UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Herramienta de predicción de corrupción y papeles de Panamá

Autor: Jorge Gallego González
Tutor: David Renato Domínguez Carreta

Junio 2018

Herremienta de predicción y los papeles de Panamá

AUTOR: Jorge Gallego González
TUTOR: David Renato Domínguez Carreta

Grupo de la EPS
Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2018

Resumen (castellano)

Este Trabajo Fin de Grado consiste en la creación de herramientas para analizar tramas de corrupción a nivel internacional, el flujo de dinero procedente de la evasión fiscal entre países e instituciones.

A este análisis se le suman herramientas de predicción que permiten conocer la evolución de la corrupción en diferentes países y regiones así como su estado o las relaciones y transaciones entre los mismos.

Este trabajo no consta de una única herramienta sino cuatro:

-Red Neuronal basada en los datos de los papeles de Panamá y los sondeos realizados por Transparencia Internacional.

Esta red neuronal tiene como primer objetivo comprobar si existe una correlación entre los datos extraídos de los papeles de Panamá y los STI y como segundo objetivo realizar predicciones sobre los niveles de corrupción de cada país en años futuros.

-Programa para analizar patrones de comportamiento respecto a corrupción de cada país:

Este programa realiza un análisis de los atributos asociados a la corrupción de cada país basandose en los resultados de estudios previos sobre los papeles de Panamá[1] para asociar los países similares entre sí y generar una matriz en la cual se refleja que países están conectados a otros pos su similitud.

-Estudio sobre conexiones locales y externas entre países:

Tomando como partida las aparicciones de cada país y las conexiones entre los mismos presentes en los papeles de Panamá este programa obtiene como resultado los enlaces entre unas regiones y otras del mundo y el peso que tienen sobre el total de conexiones así como la influencia internacional de cada país en los flujos de transacciones obserbados en los papeles de Panamá.

-Creación de un mapa de regiones:

Este programa crea una serie de regiones que son conjuntos de países, cada conjunto contiene países con comportamientos similares respecto a corrupción.

Abstract (English)

This work consist in the creation of tools to analyze corruption schemes on an international level, the cash flow from tax evasion between countries and institutions.

To this analisis are added prediction tools that allow to know the evolution of the corruption in different countries and regions as well as their state or the relations and transactions between them.

This work does not consist of one tool but four:

-Neural Net based on the data of the Panama papers and the polls made by TI

This neural net has as a main pourpose to check if there is any corelation between the data extracted from the Panama papers and the TI polls and as a second goal, to make predictions about the corruption levels of every country in future years.

-Program to analyze the behaviour paterns related to corruption in every country:

This program makes an analysis of the atributes asociated to corruption of every country based on the results of previous studies about the Panama papers[1] in order to asociate those countries that are similar and generate a matrix in where it is shown which countries are conected because of their similarities.

-Study about local and external conection between countries:

Taking as a starting point the occurrences of every country and the conections between them, this program gets as result the links between world regions and the weight this links have over the total of coneccions as well as the international influence of every country in the flow of transactions observer in the Panama papers.

-Creation of a map of regions:

This program creates a number of regions that are sets of countries, every set has countries with similar behaviours regarding the corruption.

Palabras clave (castellano)

Papeles de Panamá, Corrupción, Sistemas de predición, Red Neuronal, Análisis de Datos, Back Propagation, Cross Validation, Normalización.

Keywords (inglés)

Panama papers, Corruption, Prediction systems, Neural Networks, Data analysis, Back Propagation, Cross Validation, Normalization.

Agradecimientos

Quisiera agradecer a mis padres y a mi hermana dado que gracias a ellos he podido realizar este grado y por haberme apoyado en el transcurso del mismo.

Quisiera agradecer a mi tutor por el asesoramiento a lo largo del desarrollo del trabajo.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

La corrupción es uno de los problemas más grandes del mundo en el que vivimos, problema presente en todos los países del planeta, problema que a todos afecta y una de las principales preocupaciones políticas y económicas de la población.

En las últimas décadas la tecnología sobre analisis de datos, sistemas de modelado y sistemas de predicción ha avanzado a pasos agigantados y demostrado su utilidad en múltiples campos. Aplicar esta tecnología sobre los datos filtrados en los papeles de Panamá nos dará un mejor conocimiento sobre los movimientos de dinero defraudado y el papel que tiene cada país en este entramado.

Conocer y estudiar el problema es el primer paso para llegar a resolverlo, predecir el modo en el que se va a producir; nos permitirá evitarlo.

1.2 Objetivos

El objetivo de este trabajo es realizar un análisis de datos sobre corrupción y desarrollar una serie de herramientas que permitan comprender la situación global además de predecir la evolución del mundo respecto a este tema.

En este trabajo se trata también de asociar datos procedentes de diversas fuentes para buscar posibles sinergias.

Por último se tiene como objetivo que las herramientas creadas en el mismo sean de uso general y fáciles de comprender, para que puedan ser empleadas por futuros investigadores.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:**
  En esta parte se cuentan brevemente las motivaciones para realizar este trabajo y se plantean los objetivos, también se resume la estructura del mismo.
- **Estado del arte:**
  Se plantean las tecnologías y datos con los que se ha desarrollado el proyecto se informa también de la situación del campo en el que se desarrolla.
- **Diseño:**
  Se expone el diseño de las herramientas a desarrollar y los analisis a realizar, se plantea también como se van a tratar los datos iniciales.
- **Desarrollo:**
  Se explica detalladamente cómo se han implementado los programas diseñados y como se han recopilado los resultados para ser interpretados.

- **Integración, pruebas y resultados:**
  Se presentan los resultados obtenidos y las correciones a en los programas o ejecuciones de los mismos, también se realiza una interpretación de los resultados.
- **Conclusiones y trabajo futuro:**
  Por último se hace una conclusión sobre los resultados obtenidos y se plantea cómo continuar una investigación en este campo y que aportaciones pueden tener las herramientas desarrolladas.

2 Estado del arte

2.1 Corrupción

La corrupción es un problema global que supone una traba en la evolución de nuestras sociedades. Lejos de ser un problema nuevo, la corrupción es tan vieja como el ser humano mismo.

El desarrollo y avance tecnológico de nuestras sociedades permite tramas de fraude fiscal cada vez más complejas. Pero estos avances tecnológicos también nos permiten luchar contra ellas.

2.1.1 Sondeos de TI

La institución Transparencia Internacional lleva desde 1993 combatiendo la corrupción a nivel global, se trata de una ONG dedicada a impulsar campañas de concienciación sobre los efectos de la corrupción, promover la adopción de reformas políticas, y establecer convenciones internacionales sobre la materia [2].

La EENI quiere reconocer públicamente el importante esfuerzo que esta organización está llevando a cabo para luchar contra la lacra de la corrupción.
Transparencia Internacional actúa tanto contra quienes corrompen como quienes se dejan corromper, tanto a nivel político como empresarial.[3]

Transparencia internacional realiza y publica estudios sobre corrupción, a los cuales se puede acceder a través de su página web.

2.1.2 Papeles de Panamá

Los papeles de panamá son once millones de documentos filtrados que datan desde el 1970 hasta 2015 que fue el año en el que fueron filtrados de forma anónima apareciendo por vez primera en el periódico
Estos documentos fueron creados por el buffete Mossack Fonseca y contienen la información de 214.488 entidades offshore. [4]

Los datos de los papeles de Panamá fueron abiertos al público por ICIJ, (International Consortium of Investigative Journalists) y estos datos junto con otras filtraciones se encuentran disponibles en su web “https://offshoreleaks.icij.org/”.

El ICIJ junto con el periódico alemán Suddeutsche Zeitung y más de 100 socios, pasaron un año analizando los documentos filtrados para exponer las posesiones offshore de líderes políticos, enlaces con escándalos globales y otros detalles de tratos financieros ocultos o fraudulentos, traficantes de droga, billionarios, celebridades y estrellas del deporte entre otros.[5]

2.1.3 Trabajos previos

Sobre las fuentes mencionadas existen ya análisis de datos importantes.
Algunas de las conclusiones de estos análisis son datos son útiles para trabajar sobre ellos y se han usado como punto de partida.

El primer trabajo es “Predicción de la corrupción vía red neuronal” realizados por Roberto Sánchez Fernandez” en Febrero de 2017. De este se han empleado las clases de corrupción.
Estas clases de corrupción son el resultado de la predicción de una red neuronal que emplea los datos de transparencia internacional y clasifica los países del mundo según si grado de corrupción.[6]

El segundo trabajo es el “Análisis de los papeles de Panamá” realizado por Miguel Mateos Robles en Junio de 2017, de este se ha empleado el “patrón dicionario”.

El patrón dicionario en una serie de vectores asignados a cada país, los valores de estos vectores se dieron en base a las relaciones entre países recopiladas de los papeles de Panamá.
Estos valores contemplan características sobre el número de apariciones que tiene un país en los papeles como “propietario”, “intermediario”, “director”, “secretario”, “presidente”. Por ejemplo un país cuyos presidentes tengan muchas apariciones en los papeles, tendra un valor alto en el atributo “presidente”.[7]
El patrón diccionario de un país representa el comportamiento de ese país respecto a corrupción.

2.2 Métodos de predicción

2.2.1 Red neuronal

Las redes neuronales están inspiradas en en modelos biológicos, son un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.

Las redes neuronales son capaces de aprender de la experiencia, de abstraer características esenciales a partir de entradas que presentan información aparentemente irrelevante y generalizar de casos anteriores a nuevos casos.

Las cualidades que las hacen valiosas son

-Aprendizaje Adaptativo: Capacidad de aprender a realizar tareas basadas en un entrenamiento o experiencia inicial.

-Auto organización: Una red neuronal se entrena de forma automatizada y crea una representación de la información que recibe mediante una etapa de aprendizaje.

-Tolerancia a fallos: La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo un gran daño.

-Operación en tiempo real:Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.[8]

2.2.2 Cross Validation

Cross Validation es un método de entrenamiento y testeo.
Consiste en dividir el conjunto de datos de entrenamiento en k grupos por ejemplo 10. Se aplica el algoritmo de entrenamiento 10 veces. En la primera ronda de entrenamiento se utiliza 9/10 (k-1/k) de los datos como entrenamiento y emplea el el 1/10 (1/k) restante para medir la precisión.
Este proceso se repite de tal forma que cada subgrupo de 1/10 (1/k) se utilizará una vez como set de validación. Cuando el proceso ha terminado se calcula la media de las 10 (k) precisiones y se toma este valor para estimar la precisión de la red. En resumen, cross-validaton proporciona una estimación de la precisión de una red neuronal construída usando valores particulares para el número de nodos ocultos y parámetros de entrenamiento.[9]

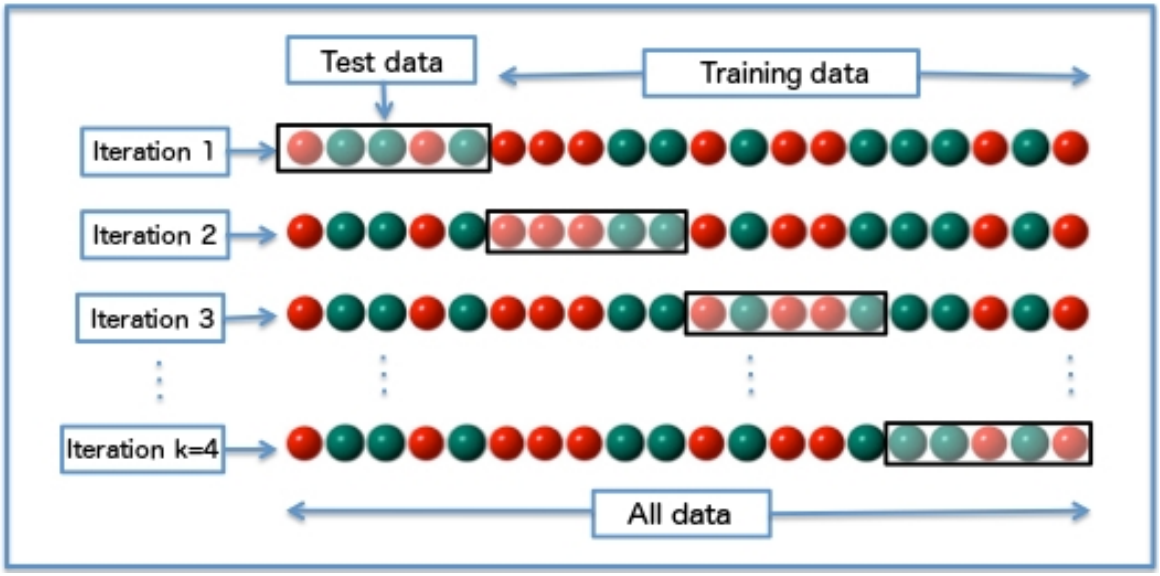


Figura 1: Cross-Validation

2.3 Rapid Miner Studio

Rapid Miner Studio es un programa de análisis de datos que permite crear, entrenar, testear y utilizar sistemas de predicción, aprendizaje automático y redes neuronales.

Rapid Miner presenta dos modos, gráfico y por consola. Por los general los procesos se realizan desde el entorno gráfico y se emplea por consola para automatizar la ejecución de procesos.

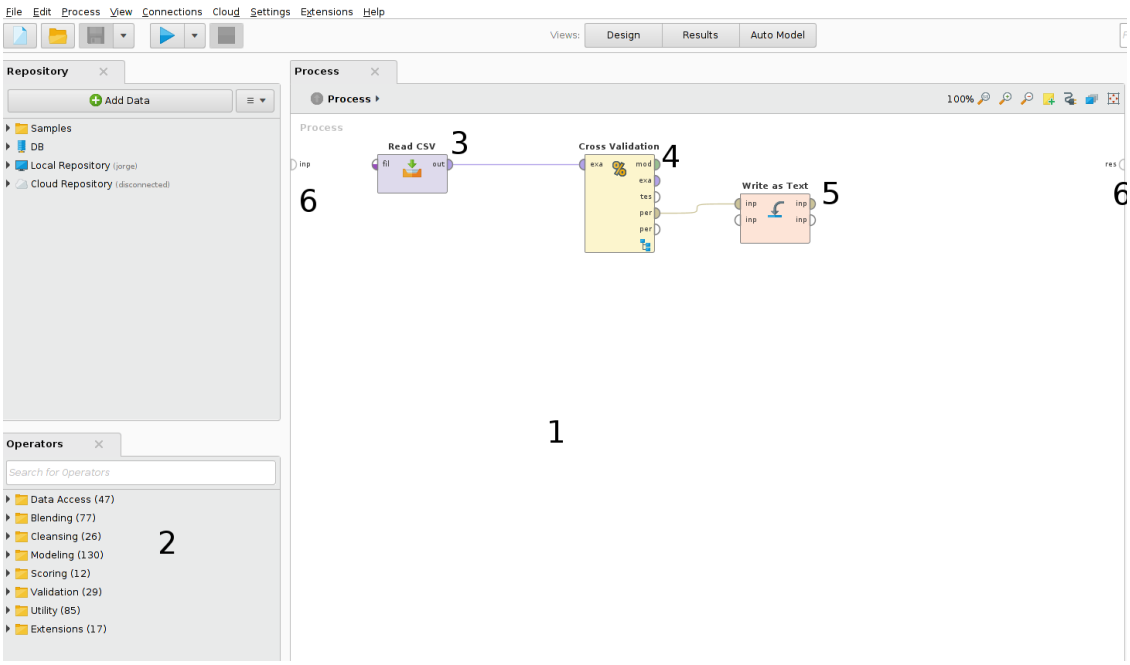


Figura 2: Interfaz RapidMiner

En la ventana “Operators” (1) tenemos los diferentes operadores que el programa pone a nuestra disposición. Estos serán los módulos que componen un proceso.
Para incorporarlos al proceso, los seleccionamos y arrastramos a la ventana “Process” (2). Una vez añadido, podemos hacer doble click en el operador para cambiar sus parámetros.

En este ejemplo vemos 3 módulo en la ventana de proceso, estos son los más habituales en procesos de clasificación, el primero (3) es un lector de fichero, permitira que nuestro sistema lea los datos de entrenamiento/test desde la ruta indicada.

El segundo (4) es el proceso de entrenamiento, haciendo doble click en este vemos su contenido. En función del proceso de entrenamiento elegido, tendrá una estructurna interna diferente. Dentro de este debemos incluir el tipo de modelo que usamos (red neuronal, árbol de decisión, perceptrón…) junto con el operador “Apply model”.

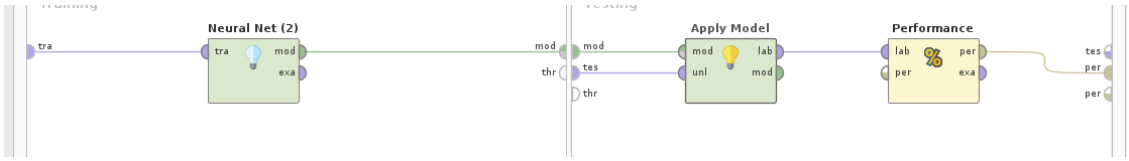


Figura 3: Interfaz RapidMiner Cross-Validation

Si queremos datos sobre la precisión del modelo, debemos añadir el operador “Performance”.

Los datos finales son escritos en un fichero utilizando el operador de escritura (5).
Si en vez de usar ficheros se desea escribir o ver los datos al momento se pueden conectar las salidas y entradas a los extremos de la ventana de proceso y tras la ejecución el programa los mostrará ddirectamente.

Una vez entrenada una red, se puede emplear para clasificar mandando nuevos datos sin clase asignada a la misma. También se puede ejecutar un proceso guardado por consola con la opción -f. (sh rapidminer-batch.sh -f $PROCESS) donde $PROCESS es la dirección a un proceso guardado en el repositorio de Rapid Miner.

Figura Estado del arte-4: Logo EPS

3 Diseño

3.1 Red neuronal

Las redes neuronales diseñadas tienen como objetivo principal buscar una correlación entre los datos procedentes de los papeles de Panamá y los datos procedentes de los STI.

Si encuentro dicha correlación puedo sacar como conclusión que los datos extraídos de los papeles de Panamá son suficientes para tener una representación del estado de corrupción global y que la perspectiva de corrupción que tiene la población sobre sus propios países es realista. (Dado que los datos de Transparencia Internacional proceden de sondeos).

El segundo objetivo, en caso de encontrar esta correlación, crear a partir de los datos extraídos de los papeles de Panamá, una red neuronal que permita predecir la evolución de un país en cuanto a corrupción en años futuros.

3.1.1 Idea principal

Para buscar una correlación se entrenará una red neuronal que tenga como entradas datos extraídos de los papeles de Panamá, y pueda clasificar un país dentro de las categorías corrupto o no corrupto, de forma que la predicción coincida con las clases extraídas del estudio sobre TI.

Cada país tiene una clasificación (Corrupto/No Corrupto...) diferente en cada año, por lo tanto para cada año se creará una red neuronal diferente.

Si encuentro un porcentaje de aciertos mayor al de un sistema de predicción aleatorio o a un sistema que prediga siempre una clase, concluiré que existe una correlación entre los datos de los papeles de Panamá y los de Transparencia Internacional, dado que los primeros permiten predecir los segundos.

3.1.2 Diseño de la red

Los datos que se van a emplear como clases son el resultado de un sistema de clasificación realizado a partir de los datos de Transparencia Internacional.

Dicho sistema de clasificación toma como entrada un conjunto de atributos que recibe cada país según los datos de TI, x_0 a x_n y predice un clase $f(x_0, \dots, x_n) = z_{TI}$.

$\vec{x}$ datos correspondientes a un país x en TI.

Así mismo, la red neuronal a construir tendrá como entrada un conjunto de atributos asignados a cada país según los datos de los papeles de Panamá (patrón diccionario[1][7]), y_0 a y_n y predice un clase $g(y_0, \dots, y_n) = z_{PP}$.

$\vec{y}$ datos correspondientes a un país x en TI.

La red buscada es tal que $z_{TI} = z_{PP}$ para todos los países.

La clasificación de países en base a los datos de TI tiene datos diferentes para cada año, por lo que se diseñará una red neuronal por cada año.

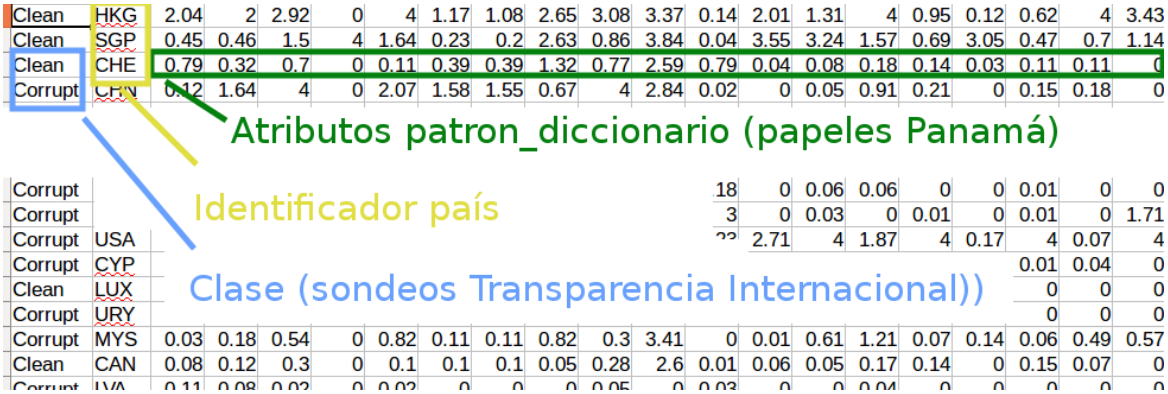


Figura 5: Set de datos de entrenamiento para la red neuronal del año 2006

Una vez obtenida esta red se compararán sus resultados de cada país con las clases asignadas para determinar si existe la correlación buscada.

En caso de encontrarse una correlación tendríamos una red neuronal para cada año (ya que las clases asignadas a los países varían cada año, la corrupción de los países varía).

Para poder entrenar una red de uso general (RNG) se empleará un set de datos de entrenamiento con los datos de todos los años.

Hay un factor más a tener en cuenta, se van a predecir resultados de años futuros, para los cuales no tenemos datos.
Para ello se entrenará la red con los mismos atributos ya empleados pero además se añadirán unos nuevos:
Tenemos datos del estado de cada país en cada año, estos se van a tomar como series temporales.
A cada caso de entrenamiento/test se le añadirán 5 atributos más, las clases de los anteriores 5 años.

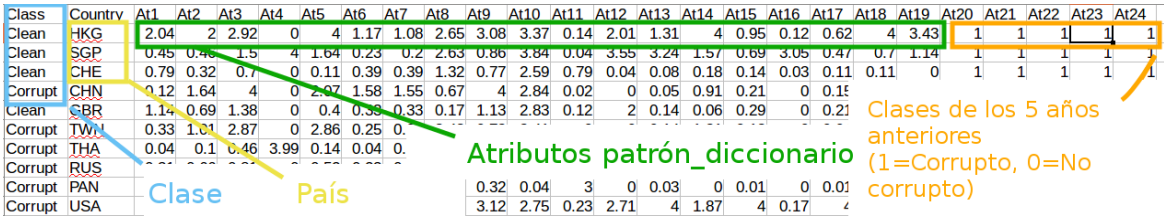


Figura 6: Set de datos de entrenamiento para la red neuronal RNG

3.2 Cálculo de similitudes entre países

Una herramienta que nos informe del grado de similitud de dos países en cuanto a comportamientos de corrupción es interesante y útil para realizar predicciones.

Estudiar la evolución de un país nos da una idea de como va a ser la evolución de los países similares a él.

Se debe tener en cuenta en este cálculo que no buscan las similitudes en grados de corrupción, sino de comportamientos de un país en cuanto a corrupción, es decir:

Si el país A es muy corrupto y según este cálculo es muy diferente al país B, eso no significaría que el país B no es corrupto, pudiera ser que el país A tiene un papel de intermediario en tramas de corrupción mientras que el B tiene una alta evasión fiscal pero no suele actuar de intermediario.

3.2.1 Cálculo de distancias

Llamaremos distancia a la diferencia entre dos países, dos países que tengan patrones de corrupción muy similares dendrán una distancia pequeña, mientras que dos países con patrones de corrupción diferentes tendrán una distancia grande.

Tomando los atributos del patrón diccionario se calculará la distancia empleando la siguiente fórmula:

Siendo
$x_0, x_1, x_2...x_n$ los atributos del patrón diccionario para el país A y
$y_0, y_1, y_2...y_n$ los atributos del patrón diccionario para el país B

la distancia d entre A y B se define de la siguiente manera:

$$d = \sum_{i=1}^{n} |xi - yi|$$

3.2.2 Normalización

Se normalizarán los resultados siguiendo la siguiente fórmula:

x = (x-min)/(max-min)

Donde x es la distacia a normalizar y min y max son la distancia mínima y máxima de todas las obtenidas.

3.2.3 Generación de la matriz de vecindad

Se generará una matriz con tantas columnas y filas como países existen.

Cada fila será un país y cada columna, su relación con otro país, en esta matriz un 0 indicará que los países son diferentes, mientras que un 1 indicará que son vecinos (similares).

Para generar esta matriz se debe establecer un umbral, y la vecindad de los países se determinará:

$_{Vab}$: Vecindad entre país A y país B

$$V_{ab} \begin{cases} V=0 \text{ si dab >umbral} \\ V=1 \text{ si dab<=umbral} \end{cases}$$

El programa dara opción a insertar varios umbrales así como mostrar la matriz con distancias normalizadas y sin normalizar.

3.3 Cálculo de conexiones locales/externas entre regiones

Este programa tomará como entradas el número de apariciones de cada pareja de países en los papeles de Panamá [11]

Estas conexiones se encuentran agrupadas por regiones, de tal forma que si existe una conexión entre el país A y el país B, A pertenece a la región x y B pertenece a la región y, se considera que existe una conexión entre la región x y la región y.

Las regiones establecidas son:
Europa Latina, Asia Oriental, Europa Germana, America Antillas, Europa Eslava, America Central, Asia Sudeste, America del Sur, Oriente Medio, America del Norte, Oceania, Africa Subsahariana, Africa Magreb y Asia del Sur.

3.3.1 Diseño del programa

El programa empezará por hacer una lista con todas las regiones, cada región tendrá los siguientes parámetros ligados a ella, nombre, conexiones locales, conexiones externas y conexiones totales.

A continuación leerá el fichero de conexiones, cada conexión encontrada puede ser de dos tipos, local o externa.

Una conexión local es aquella de una región con sí misma.
Cuando el programa lea conexiones que tienen como región entrada y región destino la misma región, las considerará locales.
Se buscará en la lista la región con el nombre de las conexiones leídas y sus parámetros se modificarán de la siguiente manera:

-Las conexiones locales leídas se añaden al número de conexiones totales de dicha región.
-Las conexiones locales leídas se añaden al número de conexiones locales de dicha región.

Una conexión externa es aquella entre dos regiones diferentes.

Cuando el programa lea conexiones que tienen como región entrada y región destino regiones diferentes, las considerará externas.

Se buscará en la lista la región con el nombre de la región entrada de la conexión y se modificarán sus parámetros de la siguiente manera:

-Las conexiones externas leídas se añaden al número de conexiones totales de dicha región.
-Las conexiones externas leídas se añaden al número de conexiones externas de dicha región.

Se hará lo mismo con la región de la lista que tenga el nombre de la región de salida.

Tras esto se calculará el número de conexiones totales sumando las conexiones totales de cada región.

Finalmente el programa realizará dos cálculos, el primero el peso (en porcentaje) de las conexiones de cada región sobre las conexiones totales que vendrá dado por:

$RP_i = 100 * RT_i / CT$

Dónde RP_i es el peso de la región i, RTi las conexiones totales de la región i y CT:

$$CT = \sum_{i}^{n} RTi$$

Siendo n el número de regiones.

El segundo valor calculado será un indicador de la influencia que tiene una región en el flujo de transaciones globales.
Este valor viene dado por:

$RI_i = 100 * RTE_i / CT$

Dónde RTE_i es el número de conexiones externas de la región i.

3.4 Mapa de regiones

Esta herramienta consistirá en un generador de mapas de regiones, dichas regiones agruparán países que tienen comportamientos similares en cuanto a corrupción.

La motivación de esto es conseguir grupos de países que se desarrollan de forma similar (en materia de evasión fiscal), de modo que las conclusiones que se puedan extraer sobre uno de los países de un grupo, se puedan extrapolar a los demás y estudiando los sucesos de un país concreto se puedan esperar comportamientos similares en los países de su mismo grupo.

Para la creación de este mapa se emplea como punto de partida la matriz de vecindad (2.3.2).

Esta matriz muestra que paises son similares entre sí, si dos países son similares la relación entre ellos se señala con un 1, si no, con un 0.

Las regiones a generar seran conjuntos que deben cumplir estas reglas:

Siendo una region R_i un conjunto de n paises $R_i=\{Px_1, Px_2...Px_n\}$,

n >= 3 Porque una región de dos países tan sólo representa similitud entre ambos, esta representación ya viene dada por la matríz y no sería útil.

$\forall$ x, y | x $\in$ {0...n}, y $\in${0...n}, y!=x

x!=f(x)

siendo f(x) = subconjunto de x

Es decir, un región no puede ser un subconjunto de otra región, esto permite reducir el número de regiones y mantener las que aportan más información.

Si por ejemplo encontramos las regiones R1 y R2.
R1 está compuesta por los países A, B y C.
R2 está compuesta por los países A, B, C y D.
R1 debe ser descartada, pues la información contenida en R1 (que los países A, B y C tienen comportamientos similares) ya está contenida en R2 y por lo tanto la región R1 es redundante.

3.4.1 Diseño del programa

Dado que esta herramienta hace uso del programa ya creado en el punto 2.3.2
se incorporará una parte a este mismo programa en lugar de hacer uno nuevo.

Se tomarán los datos sobre las distancias entre países ya calculadas y se generaran las regiones empleando el siguiente algoritmo:

3.4.2 Algoritmo principal

Este algoritmo recorrerá la lista de países buscando todas las combinaciones posibles de 3 países y en cada combinación comprobará si dichos países son vecinos entre sí, en caso de serlos, creará una región con esos tres países.

Tras esto, recorrerá toda la lista de países comprobando si cada uno pertenece a la nueva región (un país pertenece a una región si tiene conexión de vecinos con el resto de países de dicha región), insertando los que pertenezcan.

Por último se comprobará si la región creada es un subconjunto de otra ya existente, si lo es, será descartada, si no, será introducida a la lista de regiones.

Pseudocódigo:

LP = Lista de países
LR = Lista de regiones
LPx = País x, x ∈ {0...n}, n = Número de países

```
for(i=0; i<n; i++)                          /*Se recorre la lista de países*/
   R = nuevaRegion();                       /*Se crea una nueva region*/
   insertarPaísEnRegion(R, LPi);            /*Se inserta país i en la región creada*/
      for(j=i; j<n; j++)                /*Se recorren los países a partir del i actual*/
         if(i!=j)                           /*Evitamos insertar 2 veces el mismo*/
            if(pertenece(R, LPj))       /*Si el enlace x,y aparece con 1 en la matriz*/
               guardarEstadoRegion(RTempj, R)
               insertarPaísEnRegion(R, LPj);  /*Se inserta país j en la región*/

               for(k=j; k<n; k++)    /*Se recorren los países a partir de j*/
                  if((k!=j)&&(k!=i)&&(pertenece(R, LPk)))
                     guardarEstadoRegion(RTempk, R)
                     insertarPaísEnRegion(R, LPk) /*Si se han encontrado 3 conectados*/
                     If(!regionExiste(LR, R)) /*Si la region es subconjunto se descarta*/
                        for(h=0; h<n; h++)      /*Se recorren todos los países*/
                           if((h!=i)&&(h!=j)&&(h!=k)&&(pertenece(R, Lph)))
                              insertarPaísEnRegion(R, Lph)
                              restaurarEstadoRegion(RTempk, R)
                restaurarEstadoRegion(RTempk, R)
```

Es importante tener en cuenta que en los dos primeros bucles anidados j y k no recorren todos los países, j se inicializa con valor i y k con valor k.
Esto es así por optimización, de esta forma se evitan tríos redundantes.
Al buscar todas las posibles combinaciones, el orden no importa, la region R1{A, B, C} es equivalente a R2{B,C,A} y a R3{C,A,B}.

4 Desarrollo

4.1 Red Neuronal

Para la creación de las redes neuronales explicadas anteriormente, he creado los programas en C selectorYear (ficheros “selectorYear.c” y “paisClase.h”) y asignaClases (“asignaClases.c”, “paisClase.h” y “codes.h”), el script “generacionResultados.sh” y he empleado el programa RapidMiner creando un proceso usable por el mismo.

4.1.1 Programa selectorYear

Para poder realizar el diseño explicado, primero era necesario tratar los datos y dejarlos en formatos adecuados para trabajar con ellos.

Del fichero de entrada “todo_prediccion.txt”, contiene diversos valores fruto del análisis de los STI pero de estos tan sólo me interesan los campos año, país y clase asignada.

Es necesario preprocesar los datos para descartar los que no se utilizan.

Además, en este fichero hay una clase asignada para cada país en cada año, estando los datos de todos los años en el mismo fichero.

Me interesa separar estos datos de forma que tenga un fichero por año, cada uno con las clases asignadas a los países ese año.

El programa selectorYear recibe como argumento un año

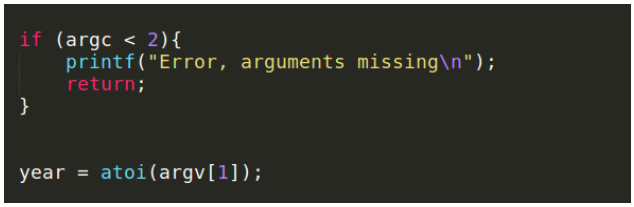


Figura 7: Comprobación Argumentos selectorYear

lee el fichero de entrada y cuando en una de las líneas leídas el año coincide con el pasado por argumento se imprimen los campos país y clase en el fichero de salida.

El fichero de salida tendrá como nombre “todo_prediccion<xxxx>.txt” siendo <xxxx> el año pasado por argumento.

```
    while ((read = getline(&line, &len, input)) != -1) {



        if (isYear(line, year)){
            counter=0;
            aux = strtok (line,"\n,");
              while (aux != NULL)
              {
                    if(counter==1){
                        fprintf (output,"%s,",aux);
                    }
                    if(counter==18){
                        fprintf (output,"%s\n",aux);
                    }
                        aux = strtok (NULL,"\n,");

                counter++;
              }


        }
    }

    fclose(input);
    fclose(output);
    return;
}


int isYear(char * line, int year){
    int y;
    char * tmpline;
    char * token;
    strcpy(tmpline, line);
    token=strtok(tmpline, ",");
    y=atoi(token);
    if(y==year)
        return 1;
    else
        return 0;

}
```

Figura 8: Código selectorYear

De esta forma queda un programa que puede generar un fichero con las clases correspondientes para cada país en un año dado. Si se le pasa un año para el cual no se tienen datos, devolverá un fichero vacío.

Ejecutando selectorYear con todos los años se descompone el fichero original en todos estos ficheros, uno por cada año.

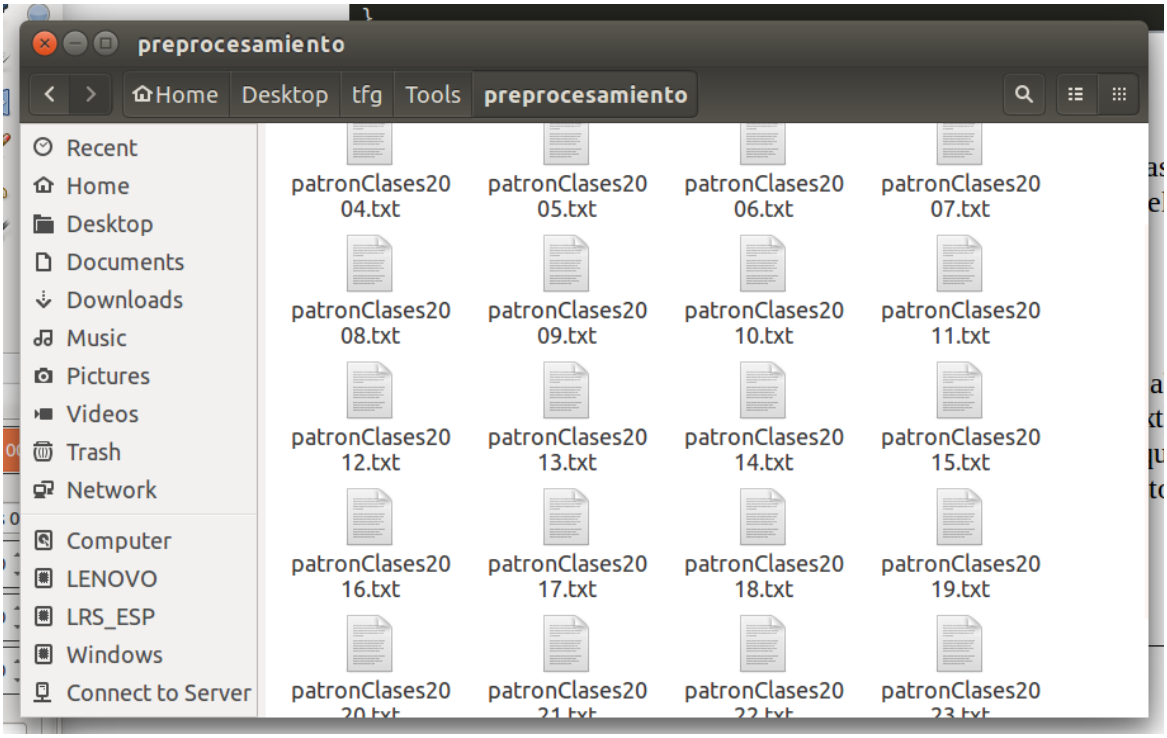


Figura 9: Ficheros generados por selectorYear

4.1.2 Programa asignaClases

El programa asignaClases (“asignaClases.c”, “paisClase.h”, “codes.h”) abre uno de los ficheros generados por selectorYear (todo_prediccion<xxxx>.txt) y el fichero “vectorPatronPaises.csv” (este contiene el patrón diccionario) que contiene los atributos asociados a cada país.
El programa recibe como argumento un año y lee el archivo correspondiente a ese año.

Este programa se divide en dos partes:

La primera parte del programa comienza intentando abrir el fichero correspondiente al año que se le ha pasado como argumento(todo_prediccion<xxxx>.txt).

```
if (argc < 2){
    printf("Error, arguments missing\n");
    return;
}


year = atoi(argv[1]);


strcat(nombrefi, nombre);
strcat(nombrefi, argv[1]);
strcat(nombrefi, nombreb);
strcat(nombrefi2, nombre2);
strcat(nombrefi2, argv[1]);
strcat(nombrefi2, nombreb2);




input = fopen(nombrefi2, "r");
if(input==NULL){
    printf("%s\n", "No hay datos de ese año");
    return;
}
```

Si no se le ha pasado argumento el programa informará del error. Si no se encuentra ningún fichero correspondiente al año pasado por argumento, el programa informará de que no hay datos para ese año.

Figura 10: Código asignaClases1

```
typedef struct paisClase paisClase;

struct paisClase{

    char nombre[MAXNOMBRE];
    char clase[MAXNOMBRE];
    paisClase * next;

};
```

Figura 11: Estructura paisClase

Tras esto el programa crea una lista enlazada de países, para ello se han utilizado la estructura paisClase, el puntero al primer elemento se utilizará como lista, el resto se irán enlazando de forma dinámica. Las funciones auxiliares sobre paisClase permiten trabajar con la lista. Estas se encuentran en "paisClase.h"

```
paisClase * crearPaisClase(); //Reserva espacio para la estructura
void inicializarPaisClase(paisClase * pais);
void destruirPaisClaseArray(paisClase * first); //Libera memoria para un pais y toda la cadena de sucesores
void enlazar(paisClase * src, paisClase * dst); //Asigna el país siguiente
char * encontrarClase(char * pais, paisClase * primero); //Recibe un nombre de pais, lo busca en la cadena de paises y
//ASIGNAR VALORES
void asignaPais(char * nombre, char * clase, paisClase * dst);
void asignaNombre(char * nombre, paisClase * dst);
void asignaClase(char * clase, paisClase * dst);
//GET VALORES
char * getNombre(paisClase * pais);
char * getClase(paisClase * pais);
paisClase * getNext(paisClase * pais);
void pushPaisClase(paisClase * pais, paisClase * first);
```

Figura 12: Funciones paisClase

El programa va leyendo el fichero todo_prediccion<xxxx>.txt por líneas y almacenando en la lista enlazada cada país con su respectiva clase.

```
while ((read = getline(&line, &len, input)) != -1) {



        aux = strtok (line,"\n,");

          while (aux != NULL)
          {

                    if(boolCount == 0){
                        tempPais = crearPaisClase();
                        asignaNombre(aux, tempPais);
                        boolCount = 1;
                    }else{
                        asignaClase(aux, tempPais);
                        pushPaisClase(tempPais , lista);
                        boolCount = 0;
                    }



                    aux = strtok (NULL,"\n,");

          }


}
```

Figura 13: Lectura de fichero asignaClases

La segunda parte del programa lee el fichero que contiene el patrón diccionario, “vectorPatronPaises.csv”.

Por cada país leído busca este país en la lista creada en la primera parte (de países y clases).

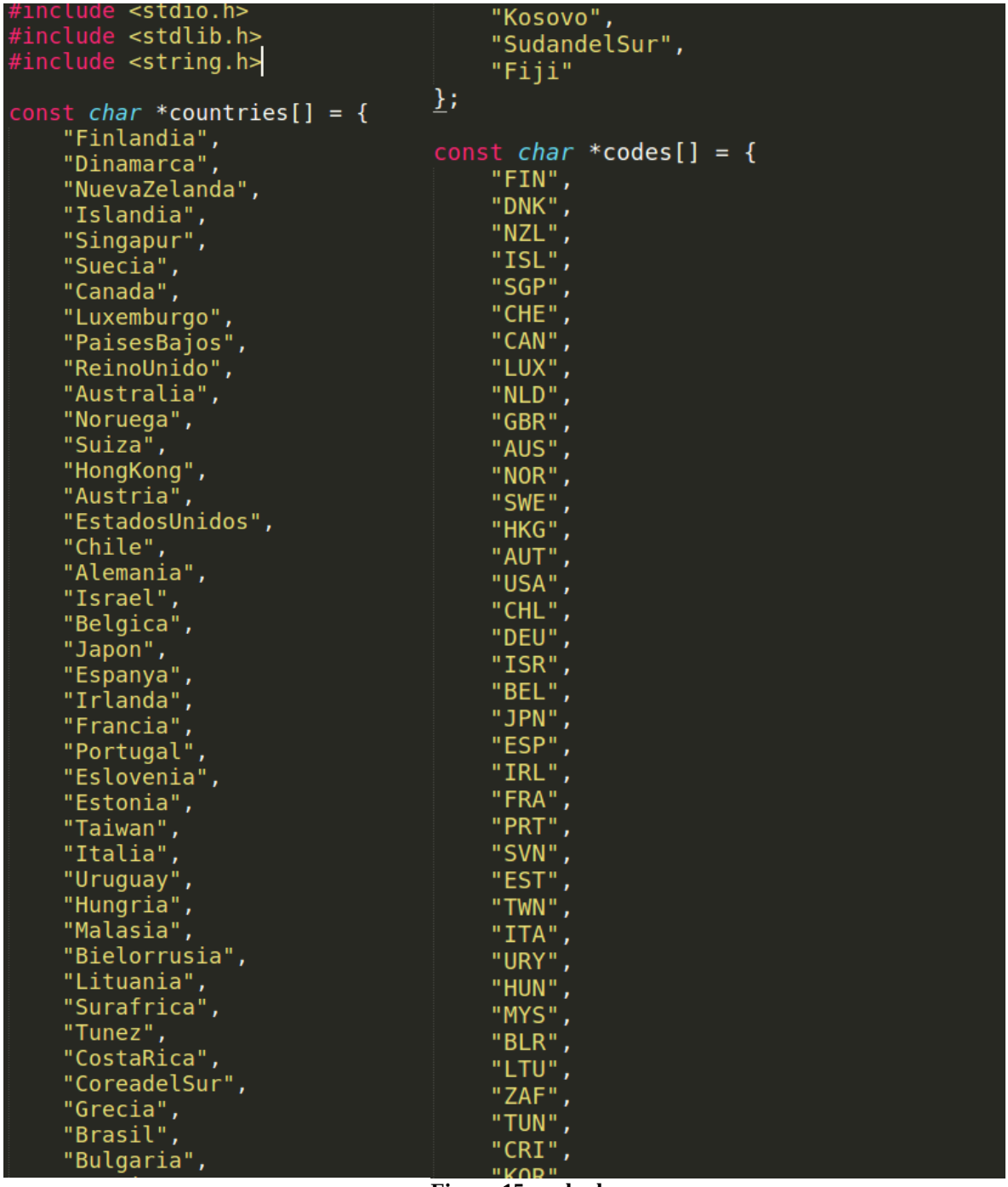
El programa finaliza escribiendo en un fichero de salida “patronClases<xxxx>” los datos del patrón diccionario pero añadiendo como campo adicional la clase asignada a cada país.

```
while ((read = getline(&line, &len, input)) != -1) {



    n=0;
    aux = strtok (line,"\n,");
      while (aux != NULL)
      {
            if(n==0){
                //Los codigos de pais no encontrados se corresponden a paises qu
                if(encontrarClase(getNombreCode(aux), lista)==NULL){
                    n=20;
                    break;
                }else{
                    strcpy(tempc, encontrarClase(getNombreCode(aux), lista));
                    //Elimino los + y - de las clases
                    removechar(tempc, '+');
                    removechar(tempc, '-');
                    fprintf(output, "%s,",tempc);
                    fprintf (output,"%s,",aux);
                }
            }
            else if(n==19){
                fprintf (output,"%s\n",aux);


            }else if (n<19){
                fprintf (output,"%s,",aux);

            }
                aux = strtok (NULL,"\n,");


        n++;
      }


}
```

Figura 14: Escritura asignaClases

Los nombres de los países no se correspondían eran diferentes en los ficheros de entrada, ya que uno utilizaba los combres completos y el otro códigos de 3 letras para identificar cada país.

Por esto mismo he creado un fichero auxiliar codes.h que contiene una estructura asociando cada país con su código, cada vez que se requiere buscar un país por código se llama a la función getNombreCode() que empleando esta estructura permite traducir códigos.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const char *countries[] = {
    "Finlandia",
    "Dinamarca",
    "NuevaZelanda",
    "Islandia",
    "Singapur",
    "Suecia",
    "Canada",
    "Luxemburgo",
    "PaisesBajos",
    "ReinoUnido",
    "Australia",
    "Noruega",
    "Suiza",
    "HongKong",
    "Austria",
    "EstadosUnidos",
    "Chile",
    "Alemania",
    "Israel",
    "Belgica",
    "Japon",
    "Espanya",
    "Irlanda",
    "Francia",
    "Portugal",
    "Eslovenia",
    "Estonia",
    "Taiwan",
    "Italia",
    "Uruguay",
    "Hungria",
    "Malasia",
    "Bielorrusia",
    "Lituania",
    "Surafrica",
    "Tunez",
    "CostaRica",
    "CoreadelSur",
    "Grecia",
    "Brasil",
    "Bulgaria",
    "Kosovo",
    "SudandelSur",
    "Fiji"
};

const char *codes[] = {
    "FIN",
    "DNK",
    "NZL",
    "ISL",
    "SGP",
    "CHE",
    "CAN",
    "LUX",
    "NLD",
    "GBR",
    "AUS",
    "NOR",
    "SWE",
    "HKG",
    "AUT",
    "USA",
    "CHL",
    "DEU",
    "ISR",
    "BEL",
    "JPN",
    "ESP",
    "IRL",
    "FRA",
    "PRT",
    "SVN",
    "EST",
    "TWN",
    "ITA",
    "URY",
    "HUN",
    "MYS",
    "BLR",
    "LTU",
    "ZAF",
    "TUN",
    "CRI",
    "KOR"
```

Figura 15: codes.h

4.1.3 RapidMiner y el proceso “TfgNeural.rpm”

Este proceso se compone de 3 módulos, “Read csv”, “Cross Validation” y “Write as a Text”.

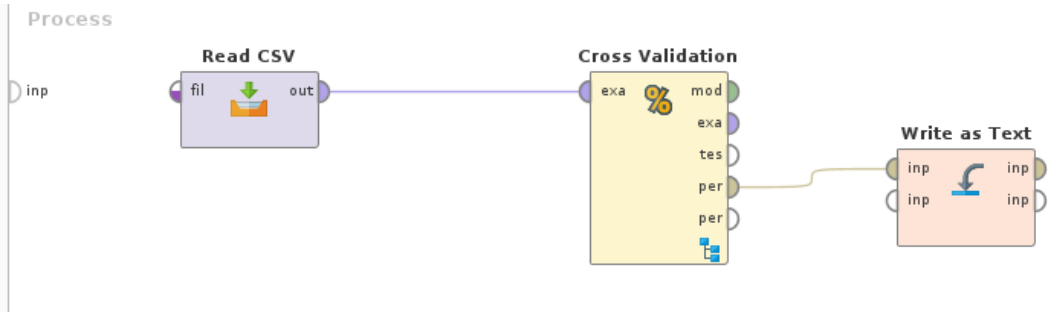


Figura 16: Proceso TfgNeural.rpm

Read CSV lee el fichero “nombrePrueba.csv” y manda los datos leídos a Cross Validation, este, contiene otros 3 módulos “Neural Net”, “Apply Model” y “Performance”.

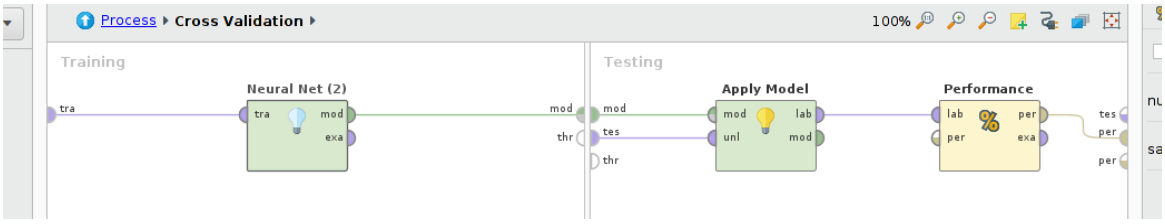


Figura 17: Proceso interno TfgNeural.rpm

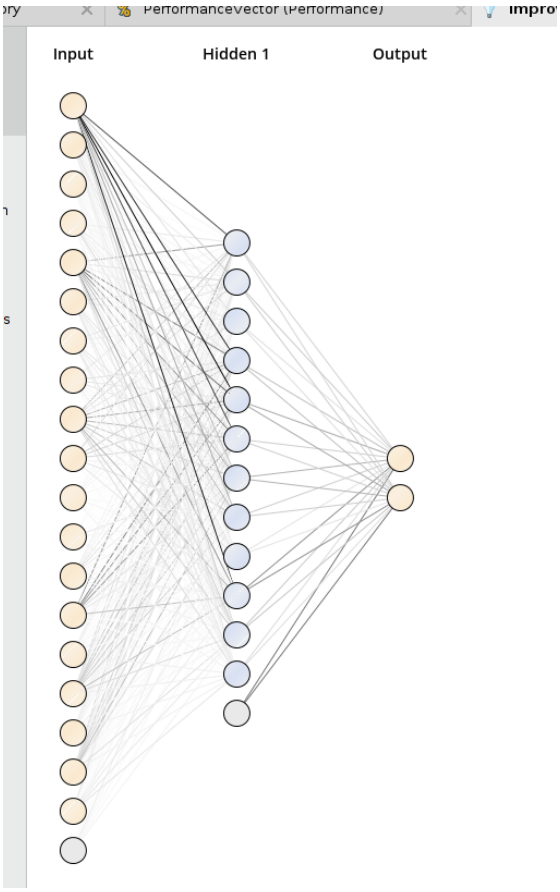


Figura 18: Red Neuronal

La red neuronal empleada se trata de una red con una capa oculta entrenada con el algoritmo de backpropagation.
Los parametros de la red se han ajustado:
Training cycles: 500
Learning rate: 0.3
Momentum: 0.2

Y el algoritmo de cross validation con k = 7.

La salida de este conjunto de módulos será el rendimiento de la red.

El módulo Write as Text genera el fichero de texto “result1.rest” con los resultados.

4.1.4 Script generacionResultados.sh

Este script combina los procesos explicados anteriormente para generar los resultados deseados de forma sencilla y automatizada.

En primer lugar, compila los programas necesarios y los mueve junto con los ficheros que van a ser leídos a una carpeta temporal.

```
gcc -o selectorYear selectorYear.c
gcc -o asignaClases asignaClases.c
gcc -o calcD2 calcularDistancias.c paisDistancia.h

mv selectorYear $TEMP
mv asignaClases $TEMP
cp todo_prediccion.txt $TEMP
cp vectorPatronPaises.csv $TEMP
```

Figura 19: Script gerenacionResultados.sh

Se puede ver en esta imagen que se están compilando más programas de los necesarios, es porque el script también ejecuta el resto de herramientas que se explicarán más adelante.

Tras preparar los programas se recorre un bucle en el que i toma valores de 2002 a 2015 (años para los que tenemos datos), en este se ejecuta en la carpeta temporal los dos programas “selectorYear” y “asignaClases” pasándoles como parámetro i.

Se modifica el nombre del fichero salida para que tenga el requerido por el proceso de RapidMiner (nombrePrueba.csv).

Se ejecuta el proceso de RapidMiner en su modo consola (sin interfaz gráfica).
Por último se modifica el nombre del resultado para tener uno más intuitivo, “prediccion i.txt” donde la i será un año.

De esta forma cada vuelta del bucle genera los resultados de la red neuronal asignada a un año. Estos resultados se guardan en el directorio /resultados

```
#Loop
for i in $(seq $STARTYEAR $ENDYEAR);
do
    cd $TEMP
    ./selectorYear $i
    ./asignaClases $i
    mv *patronClases$i* patronClases$i.txt
    mv patronClases$i.txt nombrePrueba.csv
    mv nombrePrueba.csv $TOOLS
    cd $TOOLS
    sh $LOCATION_RM -f $PROCESS
    rm nombrePrueba.csv
    mv reult1.res prediccion$i.txt
    mv prediccion$i.txt $RESULTADOS
done
#EndLoop

./calcD2 matrizDistancias

rm -R $TEMP
```

Figura 20: Bucle Script generacionResultados.sh

En el proceso se van borrando todos los ficheros temporales cuando dejan de necesitarse, tras terminar el bucle también se borra la carpeta temporal, de forma que tras la ejecución del script sólo quedan los resultados.

4.1.5 Red neuronal general

Para crear el fichero de entrenamiento de la red de uso general se han empleado los

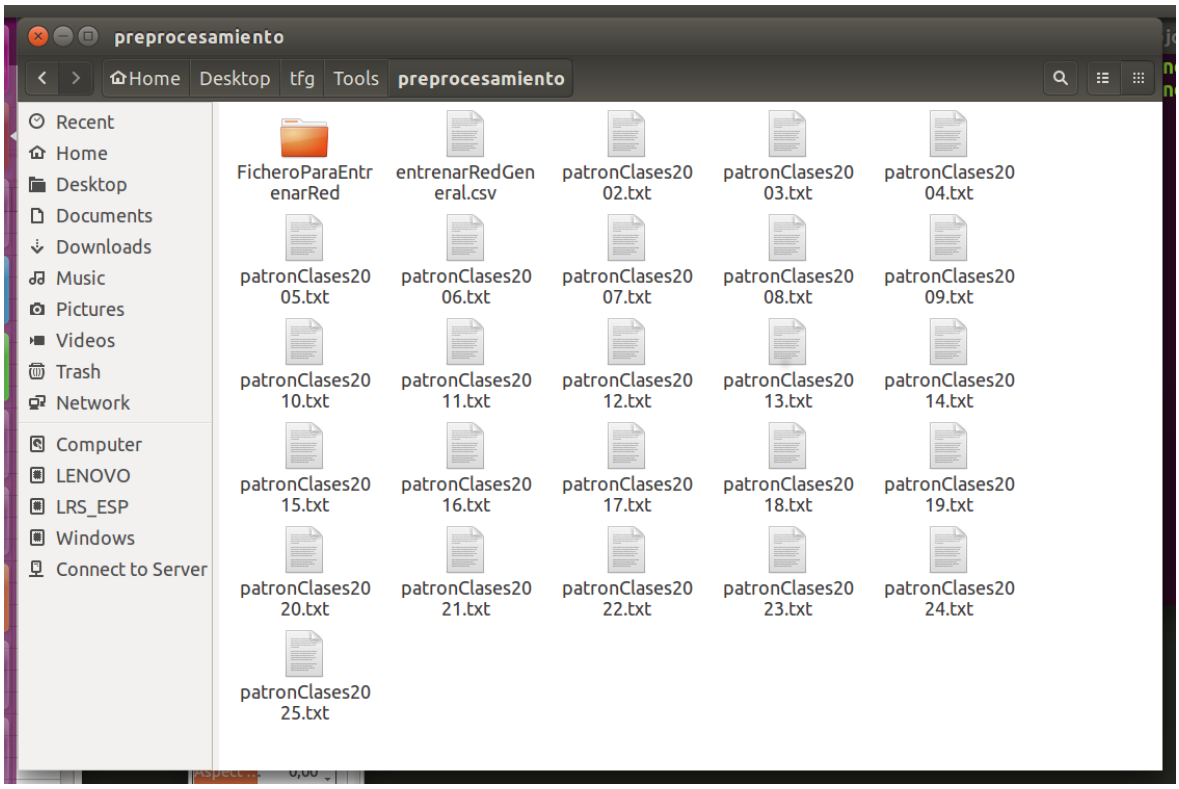


Figura 21: Ficheros de entrenamiento

ficheros generados anteriormente con los datos de cada año.
El programa redGeneral ("redGeneral.c") los combina todos en un sólo fichero y añade 6 atributos más en cada línea, el año y las clases de los 5 años anteriores (que las lee de los respectivos ficheros).

Class	Country	At1	At2	At3	At4	At5	At6	At7	At8	At9	At10	At11	At12	At13	At14	At15	At16	At17	At18	At19	At20	At21	At22	At23	At24
Clean	HKG	2.04	2	2.92	0	4	1.17	1.08	2.65	3.08	3.37	0.14	2.01	1.31	4	0.95	0.12	0.62	4	3.43	Clean	Clean	Clean	Clean	Clean
Clean	SGP	0.45	0.46	1.5	4	1.64	0.23	0.2	2.63	0.86	3.84	0.04	3.55	3.24	1.57	0.69	3.05	0.47	0.7	1.14	Clean	Clean	Clean	Clean	Clean
Clean	CHE	0.79	0.32	0.7	0	0.11	0.39	0.39	1.32	0.77	2.59	0.79	0.04	0.08	0.18	0.14	0.03	0.11	0.11	0	Clean	Clean	Clean	Clean	Clean
Corrupt	CHN	0.12	1.64	4	0	2.07	1.58	1.55	0.67	4	2.84	0.02	0	0.05	0.91	0.21	0	0.15	0.18	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Clean	GBR	1.14	0.69	1.38	0	0.4	0.33	0.33	0.17	1.13	2.83	0.12	2	0.14	0.06	0.29	0	0.21	0	0	Clean	Clean	Clean	Clean	Clean
Corrupt	TWN	0.33	1.01	2.87	0	2.86	0.25	0.24	3.48	0.73	3.41	0	0	0.14	1.31	0.13	0	0.04	0.56	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	THA	0.04	0.1	0.46	3.99	0.14	0.04	0.04	0.03	0.18	3.43	0.02	2.02	0.06	0.05	0.13	0	0.05	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	RUS	0.31	0.66	0.81	0	0.52	0.23	0.22	0.06	1.59	2.67	0.18	0	0.06	0.06	0	0	0.01	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	PAN	0.32	0.37	1.06	0	0.52	1.02	1.01	0.01	0.32	0.04	3	0	0.03	0	0.01	0	0.01	0	1.71	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	USA	0.26	0.57	0.76	0	0.32	0.15	0.14	0.2	3.12	2.75	0.23	2.71	4	1.87	4	0.17	4	0.07	4	Corrupt	Clean	Corrupt	Corrupt	Clean
Corrupt	CYP	0.4	0.31	0.52	0	0.62	0.82	0.8	0.81	0.37	0	0.07	0	0	0.03	0.01	0	0.01	0.04	0	Corrupt	Corrupt	Corrupt	Corrupt	Clean
Clean	LUX	0.35	0.09	0.2	0	0	0.09	0.09	0	0.08	0	0.1	0	0	0	0	0	0	0	0	Clean	Clean	Clean	Clean	Clean
Corrupt	URY	0.12	0.1	0.19	0	0	0.11	0.11	0	0.42	0	0.05	0	0	0	0	0	0	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	MYS	0.03	0.18	0.54	0	0.82	0.11	0.11	0.82	0.3	3.41	0	0.01	0.61	1.21	0.07	0.14	0.06	0.49	0.57	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Clean	CAN	0.08	0.12	0.3	0	0.1	0.1	0.1	0.05	0.28	2.6	0.01	0.06	0.05	0.17	0.14	0	0.15	0.07	0	Clean	Clean	Clean	Clean	Clean
Corrupt	LVA	0.11	0.08	0.02	0	0.02	0	0	0	0.05	0	0.03	0	0	0.04	0	0	0	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	BRA	0.03	0.1	0.2	0	0.01	0.08	0.08	0	0.47	0	0.01	0	0	0.01	0.01	0	0.03	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	COL	0.04	0.06	0.1	0	0	0.15	0.15	0	0.48	0	0.07	0	0	0	0	0	0	0	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Corrupt	IRL	0.08	0.09	0.03	0	0.01	0.01	0.01	0	0.04	0	0.04	0	0.02	0	0	0	0	0.04	0	Corrupt	Corrupt	Corrupt	Corrupt	Corrupt
Clean	AUS	0.01	0.11	0.4	0	0.15	0.07	0.07	0.03	0.24	3.05	0	0.32	0.09	0.67	0.07	0	0.08	0	1.14	Clean	Clean	Clean	Clean	Clean

Figura 22: Data set RNG

El tipo de red neuronal empleada será el mismo usado anteriormente.

4.2 Cálculo de similitudes entre países

La herramienta para esto es el programa calcularDistancias ("calcularDistancias.c", "paisDistancia.h") , este programa recibe como argumentos:

1. Nombre del fichero salida
2. -N (opcional) si se añade este argumento en la ejecución el programa normalizara los datos.
3. -C (opcional) si se añade este argumento en la ejecución se generará la matriz de vecindad.
4. Umbral (Obligatorio cuando se usa -C) establece el umbral usado al generar la matriz de vecindad.

El argumento 2 también puede ser -NR, esta opción se explicará más adelante.

Si hay algún tipo de error en los argumentos pasados el programa informará de cómo debe ser el formato y finalizará.

```
/*Control de errores*/
if ((argc < 2)||(argc > 5)||(argc == 4)){
    printf("Error, arguments missing or too many arguments\n");
    printf("First argument has to be the name of the output file, second is optional and has to be -N in case that you want the da
    return;
}
```

Figura 23: Control de errores calcularDistancias

El programa también informa del modo en el que se ejecuta cuando los parámetros son correctos.

```
if (argc == 3){
    if(strcmp(argv[2], "-N") == 0){
        normaliza=1;
        printf("normalize option\n");
    }else if(strcmp(argv[2], "-NR") == 0){
        normaliza=1;
        printf("normalize and generate regions option\n");
        flagOpReg=1;
    }else{
        printf("\nERROR in the arguments\n");
        return;
    }
}
```

Figura 24: calcularDistancias 2 argumentos

```
if (argc == 5){
    if((strcmp(argv[2], "-N") == 0)&&(strcmp(argv[3], "-C")==0)){

        umbral = atof(argv[4]);
        if(umbral>1||umbral<0){
            printf("Wrong threshold, should be between 0 and 1\n");
            return;
        }
        normaliza=1;
        printf("normalize & conection matrix option\n");
    }else if((strcmp(argv[2], "-NR") == 0)&&(strcmp(argv[3], "-C")==0)){
        umbral = atof(argv[4]);
        if(umbral>1||umbral<0){
            printf("Wrong threshold, should be between 0 and 1\n");
            return;
        }
        normaliza=1;
        printf("normalize, conection matrix & generate regions option\n");
        flagOpReg=1;
    }
    else{
        printf("\nERROR in the arguments\n");
        return;
    }
```

Figura 25: calcularDistancias 4 argumentos

Este programa tiene dos partes, en la primera se lee el fichero "vectorPatronPaises.csv" , cada línea leída contiene el nombre de un país y los valores del patrón diccionario asignados al mismo.

El programa crea una matriz (matrizDistancias) en la que se irán insertando los países al ser leídos.
Para ello se emplean dos estructuras:

```
struct paisDiccionario{

    char nombre[MAXNOMBRE];
    double vector[NVECTOR];

};
```

Figura 26: Estructura paisDiccionario

paisDiccionario contiene el nombre del país y un array de valores que contendrá su vector.

MatrizDistancias contiene una serie de punteros a los países diccionario, (ordenPaises) y un array bidimensional de valores (vectorDistancias).

```
struct matrizDistancias{

    paisDiccionario * ordenPaises[MAXPAIS];
    double vectorDistancias[MAXPAIS][MAXPAIS]

};
```

Figura 27: Estructura matrizDistancias

En esta matriz, cada país ocupará una posición dentro del array de punteros, esta misma posición, representará al país en la matriz, de forma que si el país A ocupa la posición 3 y el país B ocupa la posición 20 en ordenPaises , la relación entre el país A y el país B estará representada en la posición vectorDistancias[3][20].
Como la matriz es simétrica, este mismo valor estará también en vectorDistancias[20][3].

Para poder trabajar con este sistema he creado una serie de funciones auxiliares para poder trabajar con la matriz, entre otras cosas estas funciones permiten encontrar la posición de un país a partir de su nombre, extraer el valor vectorDistancias entre 2 países a partir de sus nombres y eliminar o introducir países en la matriz.

```
/*Reserva espacio para los punteros de una matrizDistancias*/
matrizDistancias * crearMatriz(matrizDistancias * matriz);
/*Introduce un pais en la posicion del array ordenPaises de la matrizDistancias matriz*/
void introducirPais(matrizDistancias * matriz, int lugar, paisDiccionario * pais);
/*Busca en el array ordenPaises de una matrizDistancias y devuelve la posicion en la que se encuentra dicho pais, si no se encue
int buscarPaisMatriz(char * pais, matrizDistancias * matriz);
/*Calcula la distancia entre los patrones diccionario de 2 paises*/
double calculaDistancia (paisDiccionario * pais1, paisDiccionario * pais2);

void introducirDistancia(char * pais1, char * pais2, double dis, matrizDistancias * matriz);
double getDistancia (char * pais1, char * pais2, matrizDistancias * matriz);
void destruirMatriz (matrizDistancias * matriz);
void introducirDistanciasPaisesDiccionario(paisDiccionario * pais1, paisDiccionario * pais2, matrizDistancias * matriz);
void setPaisDiccionarioVector(paisDiccionario * pais, double valor, int posicion);
paisDiccionario * creaPais(char * nombre);
```

Figura 28: Funciones matrizDistancias

Una vez terminado de leer el fichero, la matriz queda con los países ordenados y cada uno de ellos con su vector asignado, tras esto el programa recorrerá todos los países e ira calculando la distancia entre cada par para introducirla en la matriz.

```
/*En esta parte del programa se calculan las distancias e introducen en la matriz*/
for(i=0;i<=countPais;i++){
    for(j=i; j<=countPais; j++){
      introducirDistanciasPaisesDiccionario(matriz.ordenPaises[i], matriz.ordenPaises[j], &matriz);
    }
}
```

Figura 29: Funcion introducirDistancias

Para introducir la distancia se llama a la función introducirDistanciasPaisesDiccionario

```
void introducirDistanciasPaisesDiccionario(paisDiccionario * pais1, paisDiccionario * pais2, matrizDistancias * matriz){
    double dist;
    dist = calculaDistancia (pais1, pais2);
    introducirDistancia(pais1->nombre, pais2->nombre, dist, matriz);
}
```

Figura 30: codigo introducirDistancias

la cual llama a calcularDistancia e introduce el resultado en la matriz.

```
double calculaDistancia (paisDiccionario * pais1, paisDiccionario * pais2){
    double distanciaTotal = 0.00;
    double distanciaTemp = 0.00;
    int i = 0;

    for(i=0; i<NVECTOR; i++){
        distanciaTemp = (pais1->vector[i] - pais2->vector[i]);
        if(distanciaTemp<0){
            distanciaTemp = 0 - distanciaTemp;
        }
        distanciaTotal = distanciaTotal + distanciaTemp;
        distanciaTemp = 0.00;
    }
    return distanciaTotal;

}
```

Figura 31: Funcion calculaDistancia

La distancia se calcula siguiendo la fórmulada explicada en la parte de diseño (3.2.1).

A continuación el programa reescribe los valores de las distancias en la matriz tras normalizarlos en caso e que se ejecute el programa con la opción normalizar (-N).

```
void normalizarMatriz(int nPais, matrizDistancias * matriz){
    double min, max;
    double dist = 0;
    getMinMax(&min, &max, nPais, matriz);
    int i,j;


    for(i=0;i<=nPais;i++){
        for(j=i; j<=nPais; j++){
            dist = matriz->vectorDistancias[i][j];
            dist = (dist-min)/(max-min);
            matriz->vectorDistancias[i][j]=dist;
            matriz->vectorDistancias[j][i]=dist;
        }
    }
}
```

Figura 32: Funcion normalizarMatriz

La función de normalizado aplica la fórmula explicada en la parte de diseño 3.2.2

En la última parte del programa, este imprime en un fichero (que tendrá por nombre el pasado por argumentos + “.csv”) la lista de los países en fila y luego en columna y la matriz con las distancias entre cada pareja de países.

Si se ha ejecutado el programa con la opción de generar matriz de conexiones (-C) se generará un fichero adicional (que tendrá por nombre el nombre pasado por argumentos + “Conexiones.csv”).

Este fichero será igual que el primero generado con la diferencia de que en vez de imprimirse las distancias entre cada pareja de países, se sustituye este valor por un 0 si la distancia queda por encima del umbral establecido (los países no estarían conectados) y por un 1 en caso contrario (los países están conectados).

Ejemplos de ejecución de este programa:

```
jorge@jorge-Lenovo-YOGA-700-14ISK:~/Desktop/tfg/Tools$ gcc -o calcularDistancia
calcularDistancias.c
jorge@jorge-Lenovo-YOGA-700-14ISK:~/Desktop/tfg/Tools$ ./calcularDistancia
Error, arguments missing or too many arguments
First argument has to be the name of the output file, second is optional and has
 to be -N in case that you want the data normalized, -NR in case that you want t
he data normalized and regions generated
Last two arguments are also optional, first one should be -C and the second the
threshold between 0 and 1, it will print a file with the conections of neighbour
s
No se puede usar la opcion -C si no se usa la opcion -N
```

Figura 33: Ejecución calculaDistancia sin argumentos

Ejecutado sin argumentos, no genera ningún fichero.

```
jorge@jorge-Lenovo-YOGA-700-14ISK: ~/Desktop/tfg/Tools
jorge@jorge-Lenovo-YOGA-700-14ISK:~/Desktop/tfg/Tools$ ./calcularDistancia fiche
roDesalida -N
normalize option
jorge@jorge-Lenovo-YOGA-700-14ISK:~/Desktop/tfg/Tools$
```

Figura 34: calcularDistancia -N

Ejecutado con nombre de fichero salida “ficheroDesalida” y la opción -N
Se ha generado un fichero de salida “ficheroDesalida.csv” con las distancias.

```
jorge@jorge-Lenovo-YOGA-700-14ISK: ~/Desktop/tfg/Tools
jorge@jorge-Lenovo-YOGA-700-14ISK:~/Desktop/tfg/Tools$ ./calcularDistancia fiche
roDistancias -N -C 0.02
normalize & conection matrix option
```

Figura 35: calculaDistancia -N -C

Ejecutado con nombre de fichero salida “ficheroDistancias”, las opciones -N, -C y umbral 0.02.
Se han generado 2 ficheros de salida, “ficheroDistancias.csv” con las distancias y “ficheroDistanciasConexiones.csv” con las conexiones entre países.

4.3 Cálculo de conexiones locales/externas entre regiones

El programa calculaRegionesLocalesExternas crea una lista de regiones empleando la estructura región.

```
typedef struct region region;

struct region{

    char nombre[MAXREGION];
    int conexiones;
    int conexionesLocales;
    int conexionesExternas;
    region * next;


};
```

Figura 36: Estructura region

Esta contiene:
nombre: nombre de la región.
conexiones: conexiones totales de la región.
conexionesLocales: conexiones entre países en la misma region.
conexionesExternas: conexiones entre países de esa región y países de otras.
next: puntero a la siguiente región de la lista.

El funcionamiento es similar a la lista enlazada de países (explicada en 4.1.2), y al igual que esta, he creado una serie de funciones para trabajar con ella que pueden encontrarse en “regiones.h”.

```
region * crearRegion(); //Reserva espacio para la estructura
void inicializarRegion(region * r);
void destruirRegionArray(region * first); //Libera memoria para un r y toda la cadena de sucesores
void enlazar(region * src, region * dst); //Asigna el país siguiente
int encontrarRegion(char * r, region * primero); //Recibe un nombre de r, lo busca en la lista enlazada de primero, si se encuentra, dev

//ASIGNAR VALORES
void asignaNombre(char * nombre, region * dst);
//GET VALORES
char * getNombre(region * r);
region * getNext(region * r);
void pushRegion(region * r, region * first);

int getConexionesLocales(int posicion, region * lista);/*retorna el n de conexiones locales de la region en la posicion "posicion" de la
void setConexionesLocales(int posicion, int conexiones, region * lista);/*Modifica ell valor de la conexion local de la region en la pos
int getConexiones(int posicion, region * lista);/*retorna el n de conexiones de la region en la posicion "posicion" de la lista enlazada
void setConexiones(int posicion, int conexiones, region * lista);/*Modifica ell valor de las conexiones de la region en la posicion posi
int getConexionesExternas(int posicion, region * lista);/*retorna el n de conexiones externas de la region en la posicion "posicion" de
void setConexionesExternas(int posicion, int conexiones, region * lista);/*Modifica ell valor de las conexiones externas de la region en

/*En nombre1 se pone el nombre de la region origen y en nombre2 el nombre de la region destino de un conjunto de conexiones, esta funcio
si el origen y el destino son la misma region, se añadiran una sola vez y lo harán como locales y totales*/
void anadirConexiones(char * nombre1, char * nombre2, int conexiones, region * lista);
```

Figura 37: Funciones regiones

Estás funciones nos permiten reservar y liberar memoria para regiones, buscar regiones por su nombre o el de sus países, permiten modificar sus atributos, añadir y quitar regiones a la lista, buscar y comprobar si una región se encuentra en la lista en la lista.

El programa va leyendo el fichero con las conexiones y busca en la lista (region * lista es el puntero usado ya que apunta al primer elemento region de la lista) la existencia de las regiones leídas. Cuando existen, modifica los atributos de dichas regiones añadindo las conexiones que ha leído, si no existen, se crean e introducen en la lista y despues se añaden sus conexiones.

```
while ((read = getline(&line, &len, input)) != -1) {
        count = 0;
        aux = strtok (line,"\n,");
        count++;

          while (aux != NULL)
          {

                    if(count==1){
                        if(encontrarRegion(aux, lista)==-1){
                            tempRegion=crearRegion();
                            asignaNombre(aux, tempRegion);
                            pushRegion(tempRegion, lista);
                        }
                        strcpy(tempReg1, aux);
                    }else if(count == 2){
                        if(encontrarRegion(aux, lista)==-1){
                            tempRegion=crearRegion();
                            asignaNombre(aux, tempRegion);
                            pushRegion(tempRegion, lista);
                        }
                        strcpy(tempReg2, aux);
                    }else if(count==3){
                        printf("Llamar a anadirConexiones con %s, %s y conexiones %d\n", tempReg1, tempReg2, atoi(aux) );
                        anadirConexiones(tempReg1, tempReg2, atoi(aux), lista);
                        conexionesTotales=conexionesTotales+atoi(aux);
                    }



                    aux = strtok (NULL,"\n,");
                    count++;
          }


}
```

Figura 38: Código calculaRegionesLocalesExternas

Una vez creada la lista el programa crea el fichero “conexionesRegiones.csv” en el cual imprime una línea por cada región en la lista.

Cada línea contiene los siguientes campos, el nombre de la región, sus conexiones locales, sus conexiones externas, el peso de sus conexiones respecto a las conexiones totales y su influencia global, estas dos últimas según las fórmulas explicadas en la parte de diseño (3.3.1).

```
output = fopen("conexionesRegiones.csv", "w");
fprintf(output, "Region,Conexiones Locales,Conexiones Externas,Peso Conexiones, Influencia Global\n");
tempRegion=lista;
while(tempRegion!=NULL){
    fprintf(output, "%s,%d,%d,%.2f%%,%f%%\n",
        tempRegion->nombre,
        tempRegion->conexionesLocales,
        tempRegion->conexionesExternas,
        100*(double)tempRegion->conexiones/(double)conexionesTotales,
        100*(double)tempRegion->conexionesExternas/(double)conexionesTotales);
    tempRegion=tempRegion->next;
}

fclose(output);
destruirRegionArray(lista);
```

Figura 39: Generación del fichero salida en calculaRegionesLocalesExternas

4.4 Creación del mapa de regiones

Este proceso no se realiza en un programa nuevo, en su lugar decidí incluirlo en el programa calcularDistancias (“calcularDistancias.c”, “paisDistancia.h”) por que se va a hacer uso de algunas estructuras creadas en este programa y es más eficiente utilizarlas durante su ejecución, antes de que sean destruídas, que volver a crearlas a partir de los ficheros resultantes.

Además, tener ambos procesos en el mismo programa nos permitirá conseguir los resultados de manera más sencilla realizando una sóla ejecución.

Al programa calcularDistancias (explicado en 4.2) se le añade una opción más a pasar por argumentos, el argumento 3 puede ser -NR, en caso de pasarse este argumento el programa se ejecuta con la opción “normalizar, matriz de conexiones y generar regiones”.

Tras realizar lo ya explicado en 4.2, el programa crea una lista de regiones empleando las estructuras region y regionLista del fichero “paisClase.h”:

```
struct region{
    paisDiccionario * paises[MAXPAIS];
    int nPaises;
};
```

Figura 40: Estructura region

region contiene:paises: array punteros a paisDiccionario
nPaises: número de países en la región

```
struct regionLista{
    region * regiones[MAXREGION];
    int nRegiones;
};
```

Figura 41: Estructura regionLista

regionLista contiene:
regiones: array de regiones en la lista
nRegiones: número de regiones en la lista

A continuación, el programa implementa el algoritmo de generación de regiones explicado en la parte de diseño (3.4.1) y tomando los datos sobre conexiones entre países que se encuentran en la matriz (estructura matrizDistancias explicada en 4.2) calcula las regiones deseadas y las guarda todas en la lista.

```
for(i=0; i<countPais; i++){

    /*Se crea una nueva region*/
    regionTemp = crearRegion();

    /*Se inserta el país i*/
    insertarPaisRegion(regionTemp, matriz.ordenPaises[i]);
    regionTemp3 = *regionTemp; /*Se guarda la region con un solo pais*/

    for(j=i; j<countPais; j++){/*Recorre todos los paises emparejando pais i con aquellos que sea vecino*/

        if(i!=j){
            if(matriz.vectorDistancias[i][j]<=umbral){

                insertarPaisRegion(regionTemp, matriz.ordenPaises[j]);/*Si son vecinos se inserta, ya hay una pa
                regionTemp2 = *regionTemp; /*Guardar el valor de rregionTemp*/
                for(k=j; k<countPais; k++){/*Se recorren todos los paises para ir agrupando de 3 en 3 y consegui
                    if((k!=i)&&(k!=j)){
                        if((paisPerteneceRegion(regionTemp, matriz.ordenPaises[k], &matriz, umbral)==1)){/*Si pa
                            insertarPaisRegion(regionTemp, matriz.ordenPaises[k]); /*establecida region con pais
                            /*Comprobar si esta region es subconjunto de otra que ya esta en la lista*/
                            flagRegIn=regionPresente(&listaRegiones, regionTemp);
                        }else{
                            flagRegIn=1;
                        }
                        if(flagRegIn==0){/*Si se trata de una region nueva*/
                            /*Se recorren todos los paises insertando los que pertenecen a esta region*/
                            for(h=0; h<countPais; h++){
                                if((paisPerteneceRegion(regionTemp, matriz.ordenPaises[h], &matriz, umbral)==1)
                                    &&(h!=i)&&(h!=j)&&(h!=k)){
                                    insertarPaisRegion(regionTemp, matriz.ordenPaises[h]);
                                }
                            }
                            if(regionPresente(&listaRegiones, regionTemp)==0){
                                    /*Region terminada, ahora se inserta en la lista*/
                                infoRegionNueva(regionTemp);
                                insertarRegionLista(&listaRegiones, regionTemp);
                                /*Si la region ha sido insertada se necesita una nueva para la siguiente*/
                                regionTemp = crearRegion();
                            }


                        }/*Si la region no ha sido insertada se reutiliza para la siguiente posible region*/
```

Figura 42: Algoritmo generación de regiones en calcularDistancias

Para poder realizar este algoritmo he creado una serie de funciones que permiten trabajar con la lista de regiones, estas las he añadido al fichero paisDistancia.h.

```
/*Funciones Regiones*/
region * crearRegion();
void inicializarRegion(region * r);
void insertarPaisRegion(region * r, paisDiccionario * p);
/*devuelve 1 si el pais pertenece a la region, 0 si no*/
int paisPerteneceRegion(region * r, paisDiccionario * p, matrizDistancias * m, double umbral);
/*Devuelve 1 si las regiones son equivalentes, 0 si no*/
int regionIgual(region * r1, region * r2);
/*Devuelve 1 si la region esta presente en la lista*/
int regionPresente(regionLista * l, region * r);
void insertarRegionLista(regionLista * l, region * r);
void inicializarRegionLista(regionLista * l);
```

Figura 43: Funciones auxiliares de regiones

Estas funciones permiten crear regiones, insertar países en regiones, comprobar si un país ya está presente en una región, comprobar si una región es un subconjunto de otra, buscar una región en una lista de regiones, introducir y eliminar regiones de una lista de regiones.

Si atendemos al algoritmo de generación de regiones (explicado en 3.4.1), vemos que se requiere en algunas líneas guardar el estado de una región, pero no hay creada una función para esto.
Para guardar estos estados, he empleado algunas regiones temporales.

regionTemp es la que aparece representada en el pseudocódigo por R,
regionTemp2 y regionTemp3 se utilizan para guardar y recuperar los estados de regionTemp.

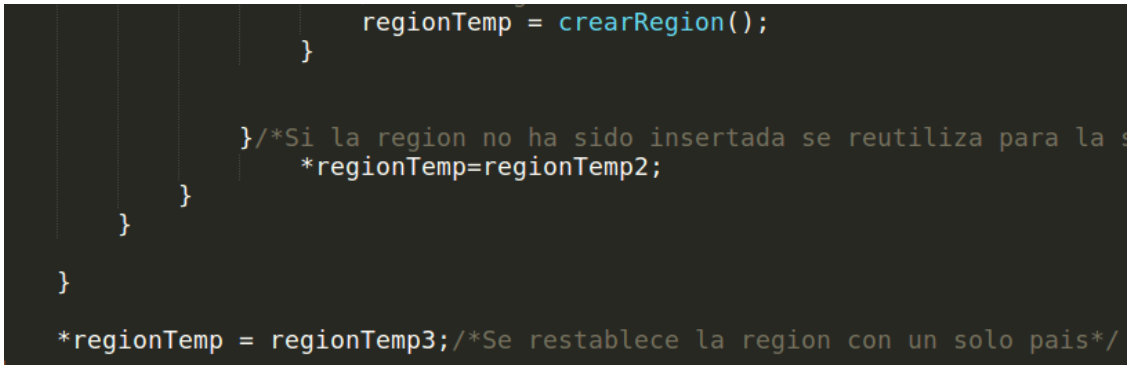


Figura 44: Guardar estados región

Una vez encontradas todas las regiones e insertadas en la lista, el programa recorre la lista imprimiendo cada región y los países que la componen en el fichero "nuevasRegiones.csv".

En cada línea se imprime:
$Region_x\{País_1, País_2...País_n\}$

5 Integración, pruebas y resultados

5.1 Red Neuronal

Tras ejecutar el script generaResultados.sh podemos observar en el directorio /resultados la precisión de cada red neuronal creada para cada año.

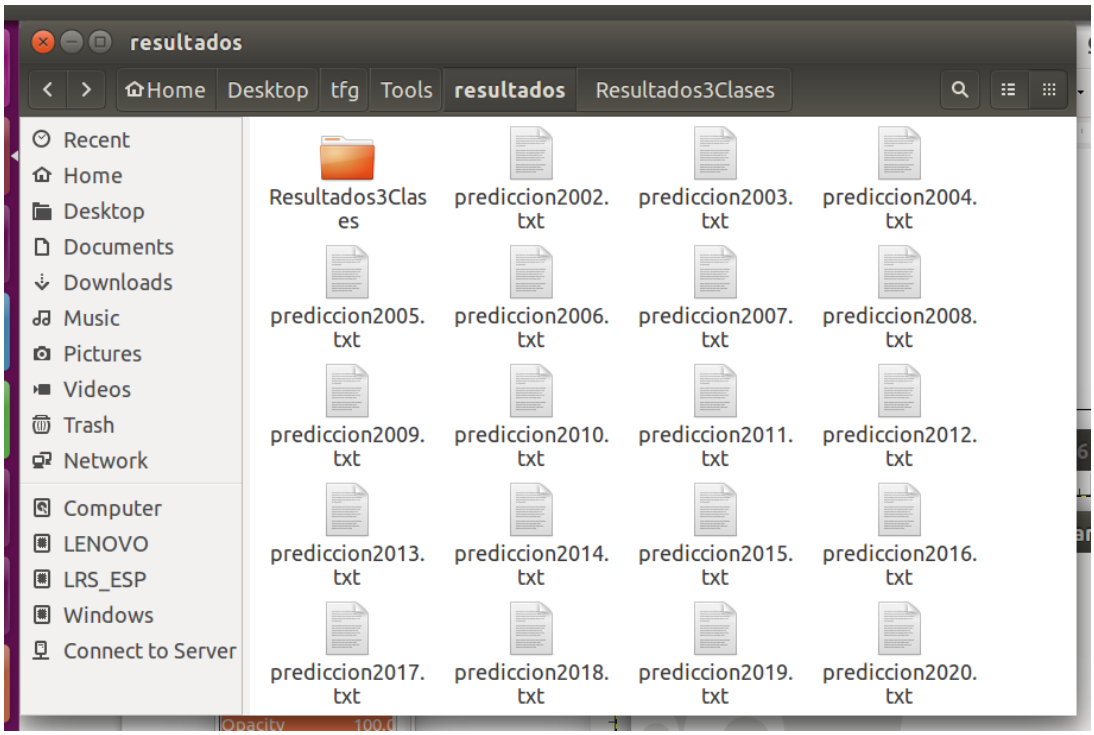


Figura 45: Directorio resultados redes neuronales

En cada fichero podemos ver los aciertos y fallos para cada clase durante el testeo y la precisión de la red.

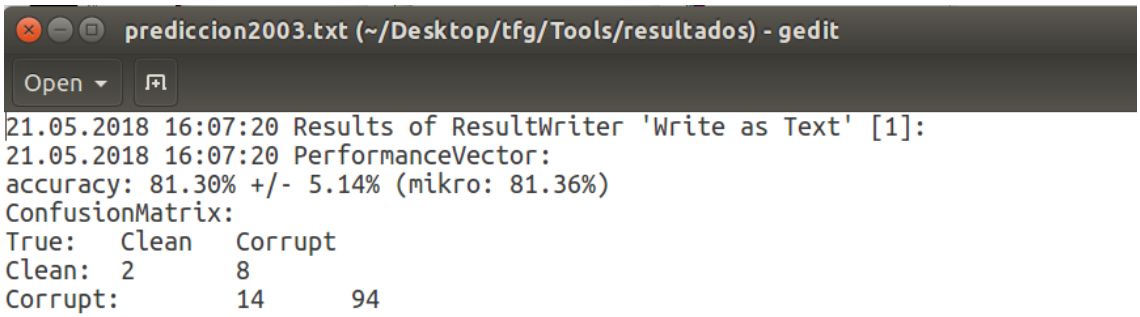


Figura 46: Precisión de la red 2003

Primero examiné los resultados con 3 clases, “Corrupto”, ”Limpio”, ”Transición” debemos recordar que para realizar estas clases, los países fueron listados de más a menos corrupto según los STI y se dividieron en 3 de forma que el tercio más corrupto quedó clasificado como “Corrupto”, el tercio menos corrupto como “Limpio” y los intermedios como “Transicción”.

Por lo tanto de acuerdo a esta clasificación, para cada año hay un tercio de países para cada clase.
Es posible cambiar el número de clases, si en vez de dividir la lista de países ordenados por corrupción en 3 se divide en 2 tendremos 2 clases, si se divide en 4, tendremos 4 clases, etc.

Para la división original de 3 clases un sistema de clasificación que predice una clase aleatoriamente (o sieempre la misma) tendría una tasa de aciertos del 33%

Tomando los datos de los ficheros resultados he creado una gráfica con la precisión de cada red neuronal (cada una asociada a un año).

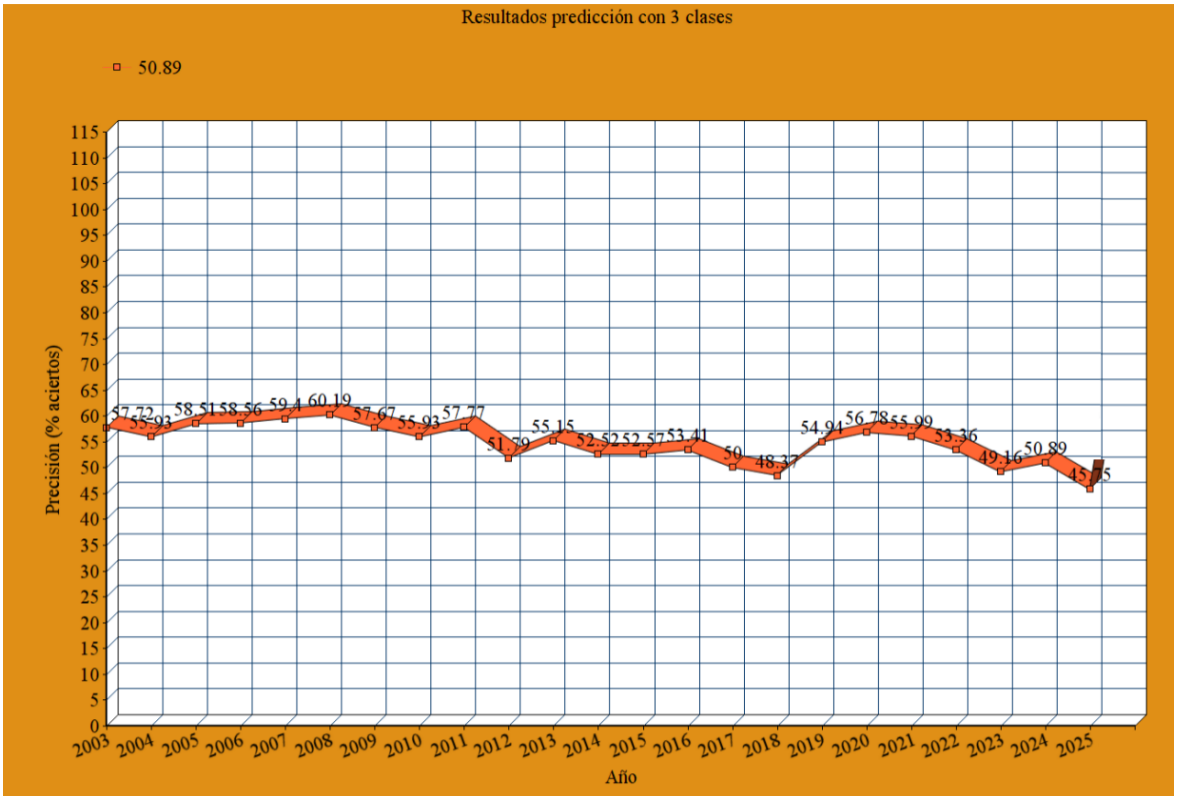


Figura 47: Resultados redes neuronales para 3 clases[12]

En esta gráfica podemos ver que los resultados se mantienen con un entre 60.19% y 48.37%, La media de la precisión de estas redes es de 56.66%.

Si lo comparamos con un sistema que haga predicciones aleatorias (o prediciendo siempre la misma clase), el porcentaje de aciertos sería del 33.33%, por lo tanto estas redes neuronales tienen un 23.33% más de aciertos. Y todas ellas superan el porcentaje de aciertos de un sistema aleatorio.

De estos resultados se puede concluir que sí existe una correlación entre los datos recabados de los papeles de Panamá y los datos proporcionados por Transparencia Internacional sobre sondeos de corrupción. Por tanto el siguiente paso es tratar de crear la red de uso general que pueda predecir la evolución de un país para años en los que no se tienen datos.

Pero antes de esto, se puede apreciar que pese a ser positivos la precisión de las redes generadas no es muy grande.

Para conseguir una red con un mejor porcentaje de aciertos, he repetido el mismo proceso pero reduciendo las clases a 2. Los resultados son los siguientes:

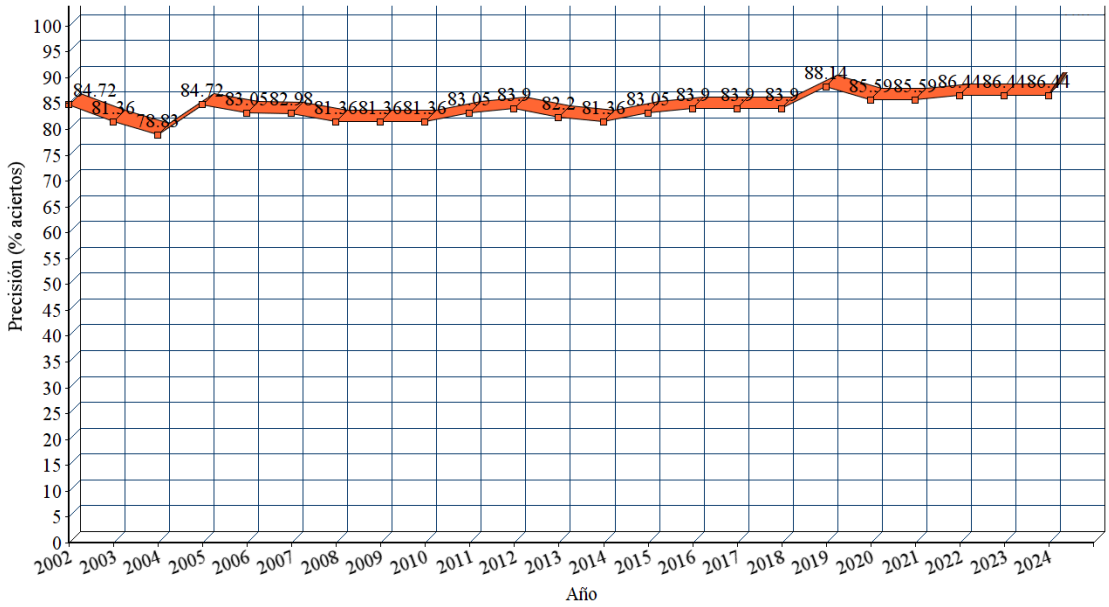


Figura 48: Resultados redes neuronales para 2 clases[12]

La precisión se mantiene entre 88.14% y 78.83% siendo la media del porcentaje de aciertos de 83.72%.
En este caso un sistema de predicción aleatorio tendría un 50% de aciertos por lo que de media estas redes lo superan en un 33.72%

Reduciendo las clases se consigue una mejoría y un sistema de predicción más fiable, no obstante la información que aporta es menor cuantas menos clases se usen, ya que las diferentes clases contienen países más diferenciados. Algunos países entre los cuales se establecía una diferencia marcandolos con clases diferentes, ahora caen dentro de la misma clase.

Por último, la red neuronal de uso general para predecir la evolución de los estados de corrupción de los países tiene los siguientes resultados:

```
20.06.2018 17:51:44 Results of ResultWriter 'Write as
Text' [1]:
20.06.2018 17:51:44 PerformanceVector:
accuracy: 98.50% +/- 0.82% (mikro: 98.50%)
ConfusionMatrix:
True:   Clean   Corrupt
Clean:  290 12
Corrupt:    22  1937
```

Figura 49: Resultados red RNG

Esta red tiene un porcentaje de fallos del 1.5%, la red es fiable para realizar este tipo de predicciones, pero se debe tener en cuenta que al ser parte de su entrenamiento basado en series temporales, este error se irá acumulando cuando tratemos calcular casos lejanos en el tiempo.

5.2 Matriz de distancias, vecinos

Aquí podemos ver una pequeña parte como muestra, se encuentran completas en los anexos.

Paises	VGB	HKG	BHS	SGP	CHE	CHN	GBR	TWN	IDN	JEY	THA	RUS	WSM	PAN	GGY	USA	CYP	LUX	ARE	CYM	URY	IMN	MYS	SYC	MCO
VGB	0	0.79	0.89	0.48	0.67	0.61	0.66	0.61	0.64	0.68	0.57	0.72	0.75	0.7	0.64	1	0.73	0.81	0.69	0.71	0.8	0.76	0.63	0.74	0.8
HKG	0.79	0	0.87	0.7	0.72	0.57	0.64	0.53	0.66	0.88	0.83	0.73	0.76	0.81	0.85	0.7	0.78	0.87	0.75	0.84	0.87	0.86	0.68	0.82	0.87
BHS	0.89	0.87	0	0.85	0.35	0.66	0.42	0.61	0.62	0.41	0.49	0.38	0.41	0.19	0.46	0.76	0.3	0.27	0.34	0.35	0.28	0.24	0.43	0.28	0.28
SGP	0.48	0.7	0.85	0	0.56	0.69	0.51	0.48	0.39	0.77	0.45	0.58	0.57	0.71	0.74	0.63	0.64	0.67	0.56	0.63	0.67	0.69	0.48	0.69	0.67
CHE	0.67	0.72	0.35	0.56	0	0.36	0.16	0.27	0.41	0.35	0.27	0.11	0.21	0.26	0.24	0.6	0.15	0.18	0.12	0.21	0.18	0.16	0.18	0.19	0.18
CHN	0.61	0.57	0.66	0.69	0.36	0	0.36	0.3	0.59	0.47	0.51	0.31	0.44	0.47	0.38	0.69	0.37	0.45	0.35	0.48	0.43	0.43	0.36	0.39	0.44
GBR	0.66	0.64	0.42	0.51	0.16	0.36	0	0.32	0.48	0.38	0.22	0.12	0.28	0.32	0.29	0.51	0.23	0.23	0.16	0.31	0.23	0.24	0.25	0.24	0.24
TWN	0.61	0.53	0.61	0.48	0.27	0.3	0.32	0	0.36	0.53	0.43	0.28	0.27	0.47	0.44	0.72	0.35	0.38	0.29	0.4	0.38	0.4	0.24	0.4	0.38
IDN	0.64	0.66	0.62	0.39	0.41	0.59	0.48	0.36	0	0.65	0.36	0.43	0.4	0.54	0.57	0.65	0.49	0.51	0.39	0.53	0.49	0.53	0.31	0.52	0.49
JEY	0.68	0.88	0.41	0.77	0.35	0.47	0.38	0.53	0.65	0	0.47	0.33	0.33	0.25	0.14	0.79	0.23	0.25	0.3	0.34	0.26	0.24	0.4	0.21	0.26
THA	0.57	0.83	0.49	0.45	0.27	0.51	0.22	0.43	0.36	0.47	0	0.24	0.35	0.41	0.34	0.64	0.31	0.25	0.21	0.32	0.24	0.27	0.25	0.28	0.23
RUS	0.72	0.73	0.38	0.58	0.11	0.31	0.12	0.28	0.43	0.33	0.24	0	0.24	0.25	0.22	0.53	0.16	0.15	0.07	0.23	0.14	0.16	0.17	0.16	0.15
WSM	0.75	0.76	0.41	0.57	0.21	0.44	0.28	0.27	0.4	0.33	0.35	0.24	0	0.28	0.34	0.7	0.17	0.16	0.21	0.17	0.16	0.18	0.24	0.17	0.15
PAN	0.7	0.81	0.19	0.71	0.26	0.47	0.32	0.47	0.54	0.25	0.41	0.25	0.28	0	0.31	0.69	0.16	0.19	0.22	0.27	0.2	0.16	0.32	0.14	0.2
GGY	0.64	0.85	0.46	0.74	0.24	0.38	0.29	0.44	0.57	0.14	0.34	0.22	0.34	0.31	0	0.73	0.24	0.25	0.2	0.32	0.25	0.23	0.29	0.2	0.24
USA	1	0.7	0.76	0.63	0.6	0.69	0.51	0.72	0.65	0.79	0.64	0.53	0.7	0.69	0.73	0	0.67	0.65	0.55	0.71	0.65	0.67	0.56	0.67	0.65
CYP	0.73	0.78	0.3	0.64	0.15	0.37	0.23	0.35	0.49	0.23	0.31	0.16	0.17	0.16	0.24	0.67	0	0.09	0.13	0.14	0.09	0.08	0.2	0.06	0.1
LUX	0.81	0.87	0.27	0.67	0.18	0.45	0.23	0.38	0.51	0.25	0.25	0.15	0.16	0.19	0.25	0.65	0.09	0	0.13	0.11	0.02	0.05	0.21	0.07	0.02
ARE	0.69	0.75	0.34	0.56	0.12	0.35	0.16	0.29	0.39	0.3	0.21	0.07	0.21	0.22	0.2	0.55	0.13	0.13	0	0.2	0.11	0.14	0.11	0.14	0.12
CYM	0.71	0.84	0.35	0.63	0.21	0.48	0.31	0.4	0.53	0.34	0.32	0.23	0.17	0.27	0.32	0.71	0.14	0.11	0.2	0	0.11	0.13	0.22	0.14	0.1
URY	0.8	0.87	0.28	0.67	0.18	0.43	0.23	0.38	0.49	0.26	0.24	0.14	0.16	0.2	0.25	0.65	0.09	0.02	0.11	0.11	0	0.05	0.2	0.07	0.02
IMN	0.76	0.86	0.24	0.69	0.16	0.43	0.24	0.4	0.53	0.24	0.27	0.16	0.18	0.16	0.23	0.67	0.08	0.05	0.14	0.13	0.05	0	0.23	0.05	0.05
MYS	0.63	0.68	0.43	0.48	0.18	0.36	0.25	0.24	0.31	0.4	0.25	0.17	0.24	0.32	0.29	0.56	0.2	0.21	0.11	0.22	0.2	0.23	0	0.23	0.2
SYC	0.74	0.82	0.28	0.69	0.19	0.39	0.24	0.4	0.52	0.21	0.28	0.16	0.17	0.14	0.2	0.67	0.06	0.07	0.14	0.14	0.07	0.05	0.23	0	0.08
MCO	0.8	0.87	0.28	0.67	0.18	0.44	0.24	0.38	0.49	0.26	0.23	0.15	0.15	0.2	0.24	0.65	0.1	0.02	0.12	0.1	0.02	0.05	0.2	0.08	0
GIB	0.78	0.85	0.27	0.7	0.19	0.42	0.25	0.4	0.53	0.23	0.27	0.17	0.18	0.17	0.22	0.68	0.07	0.04	0.14	0.13	0.04	0.03	0.23	0.04	0.04

Figura 50: Matriz distancias

Esta matriz de distancias normalizada (distancias01.csv) debe ser interpretada de la siguiente manera, cada intersección entre países representa la distancia entre estos, cuanto más cercana sea al 0, más se parecen.

En esta imagen podemos ver por ejemplo cómo Gibraltar se parece mucho a Luxemburgo con una distancia de 0.04 y en cambio es muy diferente a las Islas Vírgenes Británicas con una distancia de 0.78 o a Hong Kong con una distancia de 0.85.

Esta es una muestra de la matriz de vecindad con un umbral de 0.01 (“distanciaConexiones01.csv”):

Paises	VGB	HKG	BHS	SGP	CHE	CHN	GBR	TWN	IDN	JEY	THA	RUS	WSM	PAN	GGY	USA	CYP	LUX	ARE	CYM	URY	IMN
VGB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HKG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BHS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SGP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CHE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CHN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TWN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IDN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JEY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
THA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WSM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GGY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
USA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LUX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ARE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
URY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IMN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MYS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SYC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MCO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GIB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LVA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BRA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
COL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
LIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 51: Matriz de vecindad

Antes de estas ejecuciones he probado umbrales más altos para elaborar la matriz de vecinos, como 0.3, 0.2, 0.02.

De todos los empleados, creo que 0.01 es el más adecuado dado que umbrales más altos daban matrices con muchos 1s, esto es porque se toman para normalizar los valores máximos y mínimos.
El máximo esta muy alejado de la mayor parte del resto de los valores, al normalizar y observar la matriz de distancias, puede dar la sensación de que hay muchos países que son “similares” al estar mucho más cerca del 0 que del 1, no lo es tanto, estos valores bajos en son habituales, quedan así de bajos tras la normalización.

Para encontrar los que son similares debemos buscar un umbral muy bajo.

En esta matriz resultante de la ejecución con umbral 0.01 encontramos menos 1s, nos da una imagen más clara de que países son “vecinos”.

Las parejas de países marcadas con 1s en está matriz son países muy parecidos en cuanto a comportamientos de corrupción.

Encontramos como vecinos, Brazil y Uruguay, Colombia y Uruguay, Brazil y Colombia, Irlanda y Latvia, Perú y Uruguay, Perú y Brazil, Perú y Colombia, Jordania y Latvia, Portugal y Gibraltar, Turkía y Grecia, Polonia y Estonia, Noruega y Montenegro…

Es interesante ver como una gran parte de los países con patrones de corrupción similares tienen cierta cercanía geográfica y cultural.

5.3 Regiones nuevas

Empleando la matriz de vecindad generada, se han encontrado 7 regiones (fichero "nuevasRegones.txt"):

Región 1 {Uruguay, Brasil, Colombia, Perú}
Region 2 {Brasil, Colombia, Argentina}
Region 3 {Ecuador, Israel, Líbano, Ucrania, Arabia Saudí}
Region 4 {Israel, Argentina, Líbano}
Region 5 {Filipinas, Grecia, Corea del Sur, Kazajistán}
Region 6 {Kuwait, Bahamas, Azerbaiyán, Turkmenistán}

La última región es bastante peculiar:

Region7{Latvia, Irlanda, Jordania, Costa Rica, República Checa, Malta, Estonia, Andorra, Países Bajos, Antigua y Barbuda, Dominica, Hungría, Belgium, Chile, Macao, El Salvador, Anguilla, Egipto, Méjico, Suecia, Zimbabwe, Polonia, Viet Nam, Pakistán, Finlandia, Haití, Islandia, Curasao, Liberia, Nigeria, Bolivia, Cambodia, Polinesia Francesa, Somalia, Santa Lucía, Cuba, Jamaica, Cabo Verde, Mozambique, Albania, Nicaragua, Trinidad y Tobago, Mali, Suecia, Nepal, Nauru, Senegal, Algeria, Moldavia, Tonga, Samoa, Palestina, Lesoto, Madagascar, Puerto Rico, Tajikistán, Guinea, Yemen, Sudán, Benin, Gambia, Uganda, Martinica, Suriname, Nueva Guinea, Sierra Leona, Bosnia Herzegovina, Guam, Kirguistán, Guinea Ecuatorial, San Martín, Macedonia, Islas Nordfolk, Guyana, Guinea Bissau, Rwanda, Togo, Islas Salomón, Etiopía, Mariana Norte, Níger, Reunion, Corea del Norte, San Marino, Burkina Faso, Bhutan}

¿Realmente existe un grupo de países tan grande en el cual todos ellos son vecinos entre sí?

Si observamos la matriz "distanciaConexiones01.csv" observamos que efectivamente todos estos países aparecen indicados como "vecinos" entre sí, pero si nos vamos al origen de los datos, en el fichero con los atributos sobre estos países, muchos atributos aparecen con valor 0.

Esto es un indicador de que no se encontraron datos sobre estos países en los papeles de Panamá, efectivamente estos países tienen algo en común, sus comportamientos sobre corrupción pasan desapercibidos entre la información extraída de estos documentos, podemos decir que estos países son "invisibles" en los papeles de Panamá.

Por esto mismo los países de esta región no deben ser considerados similares en cuanto a corrupción, pero esta región si da un dato interesante, los estudios sobre los papeles de Panamá no pueden dar información relevante sobre la corrupción en estos países.

Se puede ver el mapa resultante en el anexo.

5.4 Conexiones entre regiones

Encontramos los resultados sobre el análisis de las conexiones entre regiones en el fichero "conexionesRegiones.csv".

Region	Conexiones Locales	Conexiones Externas	Peso Conexiones	Influencia Global
Europa Latina	61201	24669	17.89%	5.14%
Asia Oriental	56901	42129	20.63%	8.78%
Europa Germana	38211	36646	15.59%	7.63%
America Antillas	5216	73366	16.37%	15.28%
Europa Eslava	22283	17498	8.29%	3.64%
America Central	14131	8004	4.61%	1.67%
Asia Sudeste	7531	14810	4.65%	3.08%
America del Sur	11123	2627	2.86%	0.55%
Oriente Medio	7005	7155	2.95%	1.49%
America del Norte	6896	4826	2.44%	1.01%
Oceania	3722	9494	2.75%	1.98%
Africa Subsahariana	2185	2242	0.92%	0.47%
Africa Magreb	81	6	0.02%	0.01%
Asia del Sur	17	104	0.03%	0.02%
Total	236503	243576	100.00%	50.73%

Figura 52: Conexiones entre regiones

Podemos ver que las regiones con más presencia en los papeles de Panamá son Asia Oriental, Europa Latina, América Antillas y Europa Germana.

Es interesante comparar los pesos de las conexiones con la influencia global.
Europa latina pese a estar presente en un 17.89% de las conexiones globales tiene una influencia global relativamente pequeña, de un 5.14%, lo cual nos indica que los movimientos de fraude fiscal se producen de forma interna en su mayor parte dentro de esta región.

Asia Oriental tiene una influencia global mayor, pero también pequeña si la comparamos con el peso de sus conexiones.

America Antillas sin embargo es la región con mayor influencia global, 15.28%, muy cercana al peso de sus conexiones, 16.37%, lo cual es un indicativo de que tiene un papel de país intermediario en el flujo de corrupción global.

Podemos ver que el sumatorio de porcentajes de la "influencia global" no llega al 100% sino al 50.73%, esto es porque en este porcentaje se consideran las conexiones que no ejercen ninguna influencia global, es decir, las locales.
De este dato deducimos que aproximadamente la mitad de las conexiones presentes de los papeles de Panamá son locales y la otra mitad globales.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En el trabajo realizado se ha probado la correlación entre los papeles de Panamá y los datos proporcionados por Transparencia Internacional y se han empleado los datos conjuntamente para crear una red neuronal que nos permita predecir como van a evolucionar los niveles de corrupción de cada país con el paso de los años.

También se ha hecho un estudio de las similitudes entre países de acuerdo a los patrones que presenta cada uno en los papeles de Panamá. Se han creado herramientas para generar matrices que muestran las distancias entre países o sus similitudes.
Estas mismas herramientas permiten dibujar mapas agrupando los países en conjuntos de países "vecinos". Entendiendo por países vecinos aquellos que tienen patrones de corrupción muy parecidos.

Por último se ha realizado un programa para analizar las conexiones entre regiones del mundo de acuerdo a los datos presentes en los papeles de Panamá.
Los resultados generados nos permiten saber cuántas conexiones son locales y cuantas globales (entre regiones diferentes). Así como ver que partes del mundo están más involucradas en esquemas de fraude fiscal y que regiones tienen mayor influencia en el flujo de corrupción mundial.

6.2 Trabajo futuro

La corrupción mundial se trata de un tema extenso, es un problema muy complejo y está muy lejos de resolverse, este trabajo no está planteado con un principio y un fin definidos, sino como una parte más de una investigación que no ha terminado.

Por esto mismo todas las herramientas desarrolladas han sido pensadas con un propósito general y he tratado de documentarlas y automatizar su uso lo mejor posible para que puedan ser utilizadas en futuras investigaciones.

Las herramientas creadas pueden emplearse de diversas formas para conseguir más resultados útiles. Algunos de los resultados obtenidos dan pie a posibles estudios sobre los mismos, no obstante esto no era abarcable en un sólo trabajo.

Estudiar matrices de distancias y vecindad para diversos umbrales puedría arrojar conclusiones interesantes. También sería buena idea generar más mapas de regiones con diversos umbrales.

La red neuronal creada podría ser utilizada para generar mapas de los estados de corrupción en un futuro.

Recientemente se han filtrado los denominados "Paradise Papers", documentos con 85.000 entidades y 110.000 personalidades, IJIC los ha incorporado a su base de datos el 14 de febrero de 2018[13], realizar estudios sobre estos similares a este trabajo o los precedentes es el siguiente paso más evidente a seguir en la investigación de corrupción.

Además los datos podrían combinarse con los expuestos en este trabajo para un análisis más completo.
De realizarse un tratamiento de datos de esta nueva filtración, muchas de las herramientas desarrolladas en este trabajo se podrían utilizar sobre esos datos.

Referencias

[1] Miguel Mateo Robles. “Análisis de datos de los papeles de Panamá utilizando métodos de IA” Junio, 2017.

[2] ¿Que es Transparencia Internacional?” https://transparencia.org.es/que-es-ti/ Consultado el 18 de junio de 2018.

[3] EENI- Escuela de Negocios & Universidad Hispano-Africana de Negocios Internacionales (1995-2018) “Sobre Transparencia Internacional” http://www.reingex.com/Transparencia-Internacional.shtml Consultado el 12 de mayo de 2018.

[4] Garside, Juliette; Watt, Holly; Pegg, David (3 abril, 2016). “The Panama Papers: how the world’s rich and famous hide their money offshore”. The Guardian. Archivado del original el 3 de abril, 2016.

[5] Michael Hudson, Martha M. Hamilton, Gerard Ryle, Marina Walker Guevara and Tom Stites. “International Consortium of Investigative Journalists, Investigations, The Panama Papers”. https://www.icij.org/investigations/panama-papers/pages/panama-papers-about-the-investigation/ (Abril 2, 2018).

[6] Roberto Sánchez Fernandez “Predicción de la corrupción vía red neuronal” Febrero, 2017, pp. 45.

[7] Miguel Mateo Robles. “Análisis de datos de los papeles de Panamá utilizando métodos de IA” Junio, 2017, pp. 14-30.

[8] Hugo Galán Asensio, Alexandra Martínez Bowen. “Inteligencia artificial, redes neuronales y aplicaciones”. UCM.

[9] James McCaffrey. “Understanding and Using K-Fold Cross-Validation for Neural Networks”. Visual Studio Magazine. (24 octubre, 2013).

[10] Imagen de “https://raw.githubusercontent.com/qingkaikong/blog/master/2017_05_More_on_applying_ANN/figures/figure_1.jpg”

[11] Miguel Mateo Robles. “Análisis de datos de los papeles de Panamá utilizando métodos de IA” Junio, 2017, pp. 34-35.

[12] Imagen generada usando la herramienta online “https://www.onlinecharttool.com/”

[13] Miguel Fiandor Gutiérrez, Pierre Romera, Rigoberto Carvajal, Emilia Díaz-Struck, Delphine Reuter and Manuel Villa. “International Consortium of Investigative Journalists, Investigations, The Panama Papers”. https://www.icij.org/investigations/paradise-papers/data-ever-added-offshore-leaks-database/ (Abril 2, 2018).

Glosario

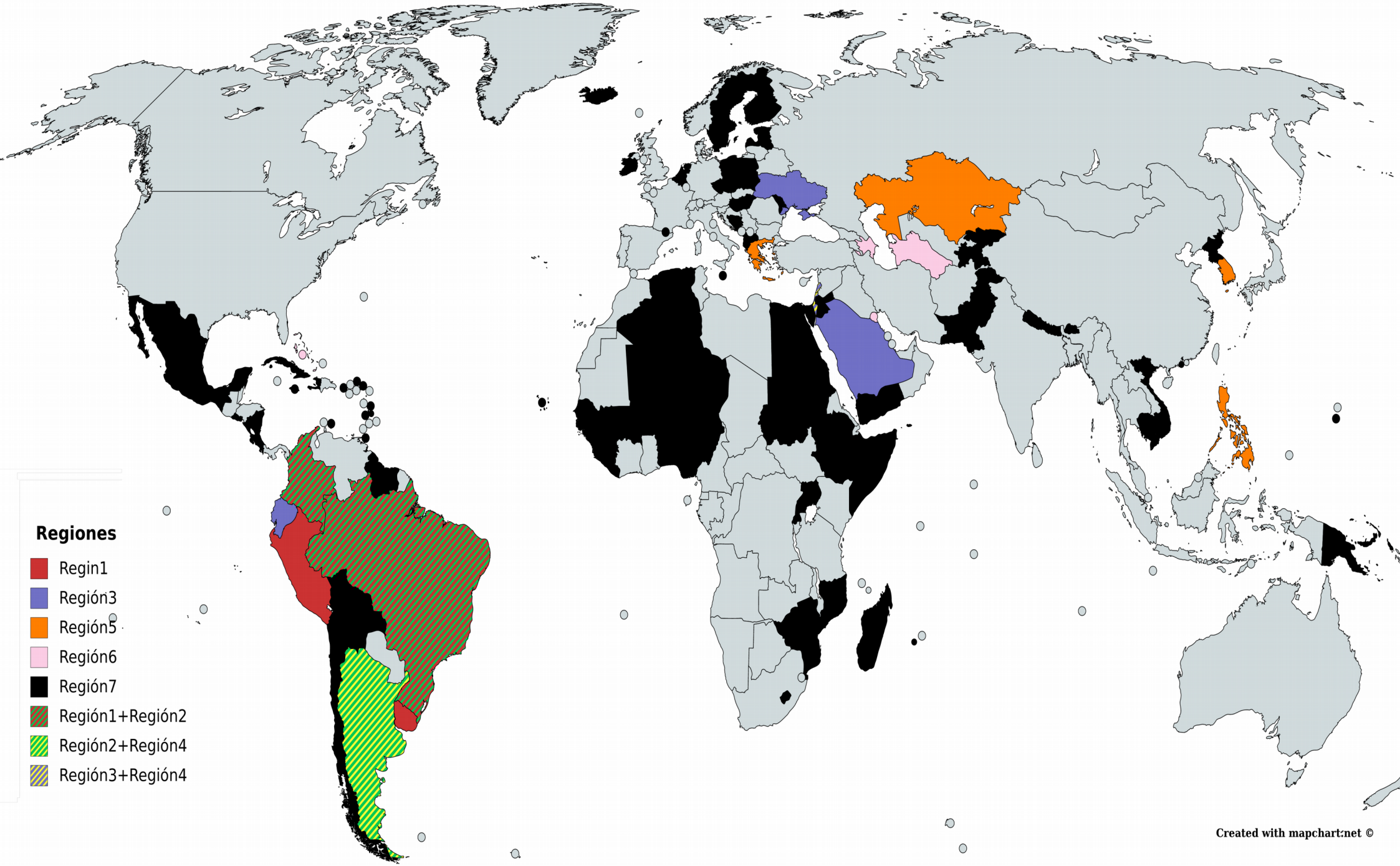
TI	Transparencia Internacional
STI	Sondeos de Transparencia Internacional
ICIJ	International Consortium of Investigative Journalists
RNG	Red Neuronal General

Anexos

Anexo 1: Mapa de regiones

En el se muestran las regiones:

Región 1 {Uruguay, Brasil, Colombia, Perú}
Region 2 {Brasil, Colombia, Argentina}
Region 3 {Ecuador, Israel, Líbano, Ucrania, Arabia Saudí}
Region 4 {Israel, Argentina, Líbano}
Region 5 {Filipinas, Grecia, Corea del Sur, Kazajistán}
Region 6 {Kuwait, Bahamas, Azerbaiyán, Turkmenistán}
Region7{Latvia, Irlanda, Jordania, Costa Rica, República Checa, Malta, Estonia, Andorra, Países Bajos, Antigua y Barbuda, Dominica, Hungría, Belgium, Chile, Macao, El Salvador, Anguilla, Egipto, Méjico, Suecia, Zimbabwe, Polonia, Viet Nam, Pakistán, Finlandia, Haití, Islandia, Curasao, Liberia, Nigeria, Bolivia, Cambodia, Polinesia Francesa, Somalia, Santa Lucía, Cuba, Jamaica, Cabo Verde, Mozambique, Albania, Nicaragua, Trinidad y Tobago, Mali, Suecia, Nepal, Nauru, Senegal, Algeria, Moldavia, Tonga, Samoa, Palestina, Lesoto, Madagascar, Puerto Rico, Tajikistán, Guinea, Yemen, Sudán, Benin, Gambia, Uganda, Martinica, Suriname, Nueva Guinea, Sierra Leona, Bosnia Herzegovina, Guam, Kirguistán, Guinea Ecuatorial, San Martín, Macedonia, Islas Nordfolk, Guyana, Guinea Bissau, Rwanda, Togo, Islas Salomón, Etiopía, Mariana Norte, Níger, Reunion, Corea del Norte, San Marino, Burkina Faso, Bhutan}


Regiones
Regin1
Región3
Región5
Región6
Región7
Región1+Región2
Región2+Región4
Región3+Región4
Created with mapchart:net ©

Anexo 2: Matriz de distancias

Las matrices de distancias y de vecindad son muy grandes, exponerlas aquí ocupa 40 páginas y se vuelven incomprensibles en este formato, adjunto un enlace a drive dónde pueden verse completas y me limito a poner aquí un par de páginas de referencia:

https://drive.google.com/open?id=1NxV0rfgjzBN5HO8PwhSr8bP2Fv8FyBwr

distancias01

Paises	VGB	HKG	BHS	SGP	CHE	CHN	GBR	TWN	IDN	JEY	THA	RUS	WSM	PAN	GGY	USA
VGB	0	0.79	0.89	0.48	0.67	0.61	0.66	0.61	0.64	0.68	0.57	0.72	0.75	0.7	0.64	1
HKG	0.79	0	0.87	0.7	0.72	0.57	0.64	0.53	0.66	0.88	0.83	0.73	0.76	0.81	0.85	0.7
BHS	0.89	0.87	0	0.85	0.35	0.66	0.42	0.61	0.62	0.41	0.49	0.38	0.41	0.19	0.46	0.76
SGP	0.48	0.7	0.85	0	0.56	0.69	0.51	0.48	0.39	0.77	0.45	0.58	0.57	0.71	0.74	0.63
CHE	0.67	0.72	0.35	0.56	0	0.36	0.16	0.27	0.41	0.35	0.27	0.11	0.21	0.26	0.24	0.6
CHN	0.61	0.57	0.66	0.69	0.36	0	0.36	0.3	0.59	0.47	0.51	0.31	0.44	0.47	0.38	0.69
GBR	0.66	0.64	0.42	0.51	0.16	0.36	0	0.32	0.48	0.38	0.22	0.12	0.28	0.32	0.29	0.51
TWN	0.61	0.53	0.61	0.48	0.27	0.3	0.32	0	0.36	0.53	0.43	0.28	0.27	0.47	0.44	0.72
IDN	0.64	0.66	0.62	0.39	0.41	0.59	0.48	0.36	0	0.65	0.36	0.43	0.4	0.54	0.57	0.65
JEY	0.68	0.88	0.41	0.77	0.35	0.47	0.38	0.53	0.65	0	0.47	0.33	0.33	0.25	0.14	0.79
THA	0.57	0.83	0.49	0.45	0.27	0.51	0.22	0.43	0.36	0.47	0	0.24	0.35	0.41	0.34	0.64
RUS	0.72	0.73	0.38	0.58	0.11	0.31	0.12	0.28	0.43	0.33	0.24	0	0.24	0.25	0.22	0.53
WSM	0.75	0.76	0.41	0.57	0.21	0.44	0.28	0.27	0.4	0.33	0.35	0.24	0	0.28	0.34	0.7
PAN	0.7	0.81	0.19	0.71	0.26	0.47	0.32	0.47	0.54	0.25	0.41	0.25	0.28	0	0.31	0.69
GGY	0.64	0.85	0.46	0.74	0.24	0.38	0.29	0.44	0.57	0.14	0.34	0.22	0.34	0.31	0	0.73
USA	1	0.7	0.76	0.63	0.6	0.69	0.51	0.72	0.65	0.79	0.64	0.53	0.7	0.69	0.73	0
CYP	0.73	0.78	0.3	0.64	0.15	0.37	0.23	0.35	0.49	0.23	0.31	0.16	0.17	0.16	0.24	0.67
LUX	0.81	0.87	0.27	0.67	0.18	0.45	0.23	0.38	0.51	0.25	0.25	0.15	0.16	0.19	0.25	0.65
ARE	0.69	0.75	0.34	0.56	0.12	0.35	0.16	0.29	0.39	0.3	0.21	0.07	0.21	0.22	0.2	0.55
CYM	0.71	0.84	0.35	0.63	0.21	0.48	0.31	0.4	0.53	0.34	0.32	0.23	0.17	0.27	0.32	0.71
URY	0.8	0.87	0.28	0.67	0.18	0.43	0.23	0.38	0.49	0.26	0.24	0.14	0.16	0.2	0.25	0.65
IMN	0.76	0.86	0.24	0.69	0.16	0.43	0.24	0.4	0.53	0.24	0.27	0.16	0.18	0.16	0.23	0.67
MYS	0.63	0.68	0.43	0.48	0.18	0.36	0.25	0.24	0.31	0.4	0.25	0.17	0.24	0.32	0.29	0.56
SYC	0.74	0.82	0.28	0.69	0.19	0.39	0.24	0.4	0.52	0.21	0.28	0.16	0.17	0.14	0.2	0.67
MCO	0.8	0.87	0.28	0.67	0.18	0.44	0.24	0.38	0.49	0.26	0.23	0.15	0.15	0.2	0.24	0.65
GIB	0.78	0.85	0.27	0.7	0.19	0.42	0.25	0.4	0.53	0.23	0.27	0.17	0.18	0.17	0.22	0.68
MUS	0.74	0.81	0.33	0.61	0.12	0.38	0.18	0.32	0.44	0.31	0.18	0.09	0.2	0.26	0.18	0.59
CAN	0.72	0.79	0.35	0.59	0.11	0.36	0.17	0.31	0.42	0.32	0.17	0.09	0.2	0.26	0.19	0.57
LVA	0.81	0.88	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.26	0.24	0.16	0.16	0.21	0.25	0.66
BRA	0.8	0.87	0.29	0.67	0.18	0.44	0.23	0.38	0.49	0.26	0.24	0.15	0.16	0.2	0.25	0.65
COL	0.8	0.87	0.28	0.67	0.18	0.44	0.23	0.38	0.5	0.26	0.25	0.15	0.17	0.2	0.25	0.65
LIE	0.79	0.87	0.27	0.68	0.18	0.43	0.23	0.38	0.51	0.25	0.25	0.15	0.16	0.19	0.24	0.66
COK	0.71	0.78	0.42	0.51	0.34	0.6	0.35	0.51	0.5	0.45	0.37	0.38	0.28	0.36	0.48	0.59
IRL	0.81	0.88	0.29	0.69	0.19	0.45	0.25	0.39	0.51	0.27	0.24	0.16	0.15	0.21	0.25	0.66
AUS	0.67	0.74	0.34	0.55	0.17	0.39	0.2	0.32	0.38	0.32	0.19	0.14	0.24	0.26	0.25	0.54
ECU	0.8	0.88	0.28	0.68	0.19	0.45	0.24	0.39	0.5	0.26	0.24	0.16	0.16	0.2	0.25	0.66
IND	0.67	0.75	0.37	0.55	0.17	0.37	0.2	0.29	0.38	0.35	0.19	0.13	0.24	0.28	0.25	0.55
ESP	0.74	0.77	0.29	0.62	0.18	0.44	0.24	0.38	0.38	0.32	0.23	0.15	0.26	0.23	0.24	0.55
ZAF	0.79	0.86	0.28	0.67	0.18	0.43	0.24	0.37	0.48	0.26	0.24	0.15	0.15	0.2	0.24	0.64
ITA	0.72	0.8	0.33	0.6	0.14	0.39	0.19	0.33	0.41	0.31	0.18	0.11	0.23	0.25	0.21	0.58
PER	0.8	0.87	0.28	0.67	0.18	0.43	0.23	0.37	0.49	0.26	0.25	0.14	0.17	0.2	0.25	0.65
KNA	0.77	0.85	0.27	0.68	0.18	0.41	0.23	0.38	0.51	0.25	0.26	0.16	0.16	0.19	0.23	0.66
GTM	0.79	0.87	0.27	0.67	0.2	0.47	0.26	0.41	0.49	0.26	0.26	0.17	0.18	0.19	0.28	0.64
JPN	0.69	0.77	0.39	0.57	0.17	0.42	0.13	0.36	0.48	0.37	0.12	0.14	0.25	0.31	0.23	0.55
BLZ	0.8	0.88	0.28	0.68	0.19	0.44	0.24	0.38	0.49	0.26	0.24	0.15	0.15	0.2	0.25	0.66
ISR	0.8	0.88	0.29	0.68	0.19	0.44	0.24	0.38	0.49	0.26	0.24	0.15	0.16	0.2	0.25	0.66
PHL	0.74	0.82	0.34	0.62	0.13	0.38	0.18	0.32	0.43	0.32	0.17	0.09	0.2	0.26	0.19	0.6
VEN	0.81	0.83	0.23	0.68	0.24	0.5	0.3	0.44	0.45	0.26	0.29	0.21	0.21	0.18	0.3	0.61
VIR	0.78	0.86	0.31	0.66	0.22	0.43	0.25	0.37	0.48	0.29	0.26	0.17	0.17	0.22	0.28	0.67
JOR	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.5	0.27	0.24	0.16	0.16	0.21	0.25	0.67
TUR	0.74	0.82	0.35	0.62	0.13	0.38	0.18	0.32	0.43	0.32	0.17	0.09	0.21	0.26	0.19	0.6
KEN	0.79	0.87	0.29	0.67	0.18	0.44	0.24	0.38	0.49	0.27	0.24	0.16	0.14	0.21	0.24	0.65

DEU	0.73	0.81	0.33	0.61	0.15	0.4	0.2	0.34	0.43	0.31	0.19	0.11	0.22	0.25	0.2	0.59
ARG	0.81	0.87	0.29	0.68	0.18	0.44	0.24	0.38	0.49	0.27	0.25	0.15	0.17	0.21	0.26	0.65
PRT	0.78	0.85	0.28	0.71	0.2	0.42	0.26	0.41	0.53	0.23	0.27	0.18	0.19	0.18	0.22	0.69
FRA	0.79	0.87	0.29	0.67	0.18	0.44	0.23	0.38	0.49	0.27	0.23	0.15	0.16	0.21	0.24	0.65
GRC	0.75	0.82	0.34	0.63	0.13	0.39	0.19	0.33	0.44	0.32	0.18	0.1	0.21	0.26	0.19	0.6
CRI	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.5	0.27	0.24	0.16	0.16	0.21	0.25	0.67
LBN	0.81	0.88	0.29	0.68	0.19	0.45	0.25	0.39	0.5	0.26	0.24	0.16	0.16	0.21	0.25	0.66
KOR	0.75	0.82	0.34	0.63	0.14	0.39	0.19	0.33	0.44	0.32	0.18	0.1	0.21	0.27	0.19	0.61
UKR	0.81	0.88	0.29	0.69	0.19	0.45	0.25	0.39	0.5	0.26	0.24	0.16	0.16	0.21	0.25	0.66
CZE	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
MLT	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.16	0.16	0.21	0.25	0.67
KAZ	0.75	0.82	0.35	0.63	0.13	0.39	0.19	0.33	0.44	0.32	0.18	0.1	0.21	0.27	0.19	0.6
DOM	0.81	0.89	0.29	0.69	0.2	0.45	0.26	0.39	0.5	0.27	0.25	0.17	0.16	0.21	0.25	0.67
EST	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
NZL	0.8	0.88	0.29	0.68	0.19	0.44	0.25	0.38	0.5	0.27	0.24	0.16	0.15	0.22	0.25	0.66
LKA	0.75	0.83	0.35	0.63	0.14	0.39	0.19	0.33	0.45	0.33	0.18	0.11	0.21	0.27	0.19	0.61
SAU	0.81	0.88	0.29	0.69	0.2	0.45	0.25	0.39	0.5	0.26	0.24	0.16	0.16	0.21	0.25	0.66
DJI	0.75	0.83	0.35	0.63	0.14	0.39	0.19	0.33	0.45	0.33	0.18	0.11	0.21	0.27	0.19	0.61
VUT	0.81	0.88	0.29	0.69	0.2	0.45	0.25	0.39	0.5	0.27	0.24	0.16	0.15	0.21	0.26	0.66
AND	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
NLD	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
BMU	0.81	0.88	0.29	0.68	0.19	0.45	0.25	0.39	0.5	0.27	0.24	0.16	0.15	0.21	0.25	0.66
ATG	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
DMA	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.4	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
HUN	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
BEL	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.26	0.67
NIU	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.4	0.51	0.27	0.25	0.16	0.16	0.21	0.26	0.67
CHL	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.4	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
MAC	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.16	0.16	0.21	0.25	0.67
SLV	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
AIA	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.4	0.51	0.27	0.24	0.17	0.16	0.21	0.26	0.67
EGY	0.82	0.89	0.29	0.69	0.2	0.46	0.25	0.4	0.51	0.27	0.24	0.17	0.16	0.21	0.26	0.67
MEX	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
SWE	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
ZWE	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.25	0.67
MHL	0.81	0.89	0.29	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.25	0.17	0.16	0.21	0.25	0.67
POL	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
VNM	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
PAK	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
KWT	0.8	0.88	0.28	0.68	0.21	0.47	0.27	0.41	0.5	0.26	0.26	0.18	0.17	0.2	0.27	0.66
FIN	0.81	0.89	0.29	0.69	0.2	0.46	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.21	0.26	0.67
AUT	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
HTI	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
ISL	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
CUW	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
LBR	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
NGA	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
BOL	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
PRY	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
BHR	0.8	0.87	0.29	0.67	0.22	0.48	0.26	0.42	0.5	0.27	0.25	0.19	0.18	0.21	0.28	0.65
BGR	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.22	0.26	0.67
TCA	0.82	0.89	0.29	0.69	0.2	0.46	0.26	0.4	0.51	0.27	0.25	0.17	0.16	0.21	0.26	0.67
FJI	0.81	0.89	0.3	0.69	0.2	0.45	0.25	0.39	0.51	0.27	0.24	0.17	0.16	0.22	0.25	0.67

Anexo 3: Matriz de vecindad con umbral 0.01

Las matrices de distancias y de vecindad son muy grandes, exponerlas aquí ocupa 40 páginas y se vuelven incomprensibles en este formato, adjunto un enlace a drive dónde pueden verse completas y me limito a poner aquí un par de páginas de referencia:

https://drive.google.com/open?id=1NxV0rfgjzBN5HO8PwhSr8bP2Fv8FyBwr

distancias01Conexiones

Paises	VGB	HKG	BHS	SGP	CHE	CHN	GBR	TWN	IDN	JEY	THA	RUS	WSM	PAN	GGY
VGB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HKG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BHS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SGP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CHE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CHN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TWN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IDN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JEY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
THA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WSM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GGY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
USA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LUX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ARE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
URY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IMN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MYS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SYC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MCO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GIB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LVA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BRA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IRL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ECU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IND	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ESP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ZAF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ITA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PER	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KNA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GTM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JPN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLZ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ISR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PHL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VIR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TUR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

distancias01Conexiones

USA	CYP	LUX	ARE	CYM	URY	IMN	MYS	SYC	MCO	GIB	MUS	CAN	LVA	BRA	COL	LIE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0